МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Економічний факультет
Кафедра економіко-математичного моделювання та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА/ПРОЄКТ
на здобуття освітнього ступеня бакалавра

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ КОЛЕКТИВНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «LOCALIZ»»**

Виконав: студент 4 курсу, групи КН-4
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Пасічник Євген Сергійович

Керівник: старший викладач кафедри економіко-математичного моделювання та інформаційних технологій,
Клебан Юрій Вікторович

Рецензент: Front-end Developer "DOODLE", LLC
Місай Володимир Віталійович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри економіко-математичного моделювання та інформаційних технологій _____ (проф., д.е.н. Кривицька О.Р.)
Протокол № 11 від «18 » травня 2023 р.

Острог, 2023

Міністерство освіти і науки України
Національний університет «Острозька академія»

Факультет: *економічний*
Кафедра: *економіко-математичного моделювання та інформаційних технологій*
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри економіко-математичного моделювання
та інформаційних технологій

_____ Ольга КРИВИЦЬКА
«____» _____20__ р.

З А В Д А Н Н Я
на кваліфікаційну роботу/проєкт студента

Пасічника Євгена Сергійовича

1. Тема роботи: "Розробка вебзастосунку для колективної локалізації програмного забезпечення "Localiz"".

керівник проєкту Клебан Юрій Вікторович, старший викладач кафедри економіко-математичного моделювання та інформаційних технологій,

Затверджено наказом ректора НаУОА від 31 жовтня 2022 року №77.

2. Термін здачі студентом закінченої роботи/проєкту: 31 травня 2023 року.

3. Вихідні дані до роботи/проєкту: для створення проєкту використовувались: Visual Studio 2022 (для створення бекенд частини), Visual Studio Code (для фронтенд частини), GitHub, API MyMemory. Стек технологій: Entity Framework Core, JWT, ASP.NET Core 6, Swagger, Web API, SQL Server, C#, ReactJS, Redux, i18n, Material-UI.

4. Перелік завдань, які належить виконати: реалізувати фронтенд додатка з використанням ReactJS та Material-UI, реалізувати бекенд додатка з використанням .Net Core 6, реалізувати розпізнавання строк у файлах типу json, реалізувати чисту архітектуру, реалізувати базу даних на SQL Server.

5. Перелік графічного матеріалу: рисунки, таблиці, діаграми, лістинги.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Клебан Ю. В.	01.12.2022р.	01.12.2022р.
2	Клебан Ю. В.	01.12.2022р.	01.12.2022р.
3	Клебан Ю. В.	01.12.2022р.	01.12.2022р.

7. Дата видачі завдання: 01.12.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1.	Затвердження теми роботи/проєкту	до 31.10.2022 р.	
2.	Постановка технічного завдання	до 01.12.2022 р.	
3.	Ознайомлення з документацією	до 10.12.2022 р.	
4.	Розробка архітектури проєкту, погодження функціоналу	до 24.12.2022 р.	
5.	Розробка інтерфейсу системи	до 01.04.2023 р.	
6.	Залучення стороннього API	до 01.04.2023 р.	
7.	Тестування системи	до 01.04.2023 р.	
8.	Виправлення помилок	до 20.04.2023 р.	
9.	Попередній захист кваліфікаційної роботи/проєкту	до 31.05.2023 р.	
10.	Здача кваліфікаційної роботи/проєкту на кафедру	31.05.2023 р.	

Студент: _____ Євген ПАСІЧНИК

Керівник кваліфікаційної роботи: _____ Юрій КЛЕБАН

АНОТАЦІЯ
кваліфікаційної роботи/проєкту
на здобуття освітнього ступеня бакалавра

Тема: *Розробка вебзастосунку для колективної локалізації програмного забезпечення "Localiz"*

Автор: *Пасічник Євген Сергійович*

Науковий керівник: *Клебан Юрій Вікторович* старший викладач кафедри економіко-математичного моделювання та інформаційних технологій.

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *77 с., 35 рис., 2 табл., 22 джерел.*

Ключові слова: *локалізація, переклад, JSON, архітектура, вебдодаток, фронтенд, бекенд, API, текстовий файл.*

Короткий зміст праці:

Завданням кваліфікаційної роботи/проєкту, було розробка вебзастосунку для колективної локалізації програмного забезпечення "Localiz. Додаток повинен надавати інструменти для перекладу та локалізації тексту, також він має забезпечувати можливість колективного перекладу файлів типу JSON. Головним користувачем вебдодатку "Lokaliz" будуть локалізатори та перекладачі. Програма має надати користувачеві набір інструментів які дозволять легко локалізовувати різні додатки. Фронтенд частина створювалась задопогою React та Material-UI на javascript. Бекенд частина створювалась на .NET Core 6. При розробці дотримувались правила і рекомендації створення чистої архітектури.

The task of the qualification work/project was to develop a web application for collective localization of the Localiz. The application should provide tools for translating and localizing text, and it should also provide the ability to collectively translate JSON files. The main users of the Lokaliz web application will be localizers and translators. The program should provide the user with a set of tools that will allow them to easily localize various applications. The frontend part was created with the help of React and Material-UI in javascript. The backend part was created on .NET Core 6. The development followed the rules and recommendations for creating a clean architecture.

ЗМІСТ

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРОЦЕСУ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9
1.1. Постановка проблеми, опис базових понять бізнес-процесу 9
1.2. Аналіз продуктів-конкурентів на ринку 13
1.3 Постановка задачі 25
1.4 Організація роботи 26
Висновки до розділу 1 27
РОЗДІЛ 2 РОЗРОБКА ВЕБЗАСТОСУНКУ “LOCALIZ” 29
2.1. Опис технологічного стеку та архітектури рішення 29
2.1.1. Опис архітектури бекенду вебзастосунку “Localiz” 30
2.1.2 Опис архітектури фронтенду вебзастосунку “Localiz” 39
2.1.3 Опис реалізації вебсервісу 45
2.1.4. Сервіси серверної частини 48
2.1.5. Сервіси клієнтської частини 53
2.2. Інтеграція стороннього API 62
2.3. Перспективи розвитку та впровадження додатку на ринок вебдодатків 63
Висновки до розділу 2 65
ВИСНОВКИ 67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 70
ДОДАТКИ 72
ДОДАТОК А 72
ДОДАТОК Б 73
ДОДАТОК В 74

ДОДАТОК Г 75
ДОДАТОК Д..... 76

ВСТУП

Оскільки вебпрограми та мобільні додатки проникають у всі аспекти нашого життя більше, ніж будь-коли раніше, локалізація програмного забезпечення стала обов'язковою умовою для забезпечення бездоганної взаємодії з користувачем (UX – User Experience) у різних регіонах і культурах [10].

Якщо ви коли-небудь намагалися налаштувати мову на електронному пристрої, який був налаштований іншою мовою, ви знаєте важливість служб локалізації програмного забезпечення. Без них програма може бути в кращому випадку неприємною, а в гіршому – непридатною для використання.

Локалізація програмного продукту – приведення програмного продукту у відповідність із законами та іншими нормативно-правовими актами, стандартами, нормами і правилами, що діють в країні для якої проводиться локалізація. Локалізація передбачає лінгвістичну локалізацію, тему добре вивчену, але й може виходити далеко за її межі. Бо програмне забезпечення, розроблене в одній культурі, може мати вбудовані культурні припущення, які можуть здаватися чужими або навіть ворожими користувачам в інших культурах [18]. Зараз локалізація майже завжди передбачає локалізацію пакетного програмного забезпечення, спочатку написаного англійською мовою так як ця мова домінує на світовому ринку. Але немає жодної логічної причини, чому це має бути саме так.

Мета написання роботи полягає у здійснені проектування та розробку програмного продукта, який буде надавати інструменти та ресурси для реалізації перекладання текстового вмісту програмних додатків. Для вирішення цієї проблеми буде створено вебдодаток під назвою "Localiz". Для досягнення мети потрібно виконати ряд завдань:

- Здійснити аналіз стану локалізації проєктів на ринку;
- Спроектувати систему:
    - Архітектура Бази даних;
    - Створення діаграми USE-CASE ;
    - Архітектура самого програмно додатку, створення гнучкої системи;

- Вибір технологій, пакетів і сервісів.

- Розробити систему згідно плану та протестувати її.

Використовуватимуться на ступні методи дослідження:

- Емпіричні:
  - Експериментальний метод – використовуватиметься для вирішення проблем з додаванням нових технологій, створення власних методів для вирішення проблем локалізації та тестування;
  - Спостереження і опис – використовуватиметься для аналізу конкурентів і роботи власного програмного продукту.
- Теоретичні:
  - Аналіз – робиття всіх питань, пізнання, абстрагування його окремих сторін чи аспектів;
  - Дедукція – виведення висновку;
  - Пояснення та узагальнення;
- Системний;
- Функціональний.

Об’єкт дослідження є створення додатку для локалізації мовних пакетів комп’ютерних програм, ігор, вебсайтів, мобільних додатків.

Предметом дослідження є процес локалізації та технології та пакети які формуватимуть вебдодаток.

Отже, ми коротко оглянули потрібність локалізації та її значення. Сформулювали мету проєкту “ Localiz ”, завдання які потрібно виконати для створення проєкту та список методів для дослідження поставлених завдань і питань.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРОЦЕСУ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Постановка проблеми, опис базових понять бізнес-процесу

У сучасному глобальному світі локалізація мобільних додатків стала обов’язковою. При правильному застосуванні це допоможе деяким програмам покращити рейтинг у магазині та вийти на новий ринок.

Тепер, коли розробники мобільних, веб додатків все більше усвідомлюють ці факти, потреба в платформі локалізації мобільних додатків, яка бездоганно інтегрується з їхнім життєвим циклом розробки програмних додатків, стала обов’язковою.

Локалізація має кілька основних елементів [10]:

- Дизайн UI/UX: дизайнери адаптують інтерфейс користувача програмного продукту всіма потрібними мовами, щоб уловлювати будь-які порушення дизайну, які можуть мати негативний вплив на роботу користувача (наприклад, довжина слів у кнопках із закликом до дії).
- Переклад вмісту: одночасно перекладається вміст програмного забезпечення, а всі супутні ресурси, такі як зображення, адаптуються до вимог кожного ринку.
- Тестування та забезпечення якості (QA): нарешті, перед розгортанням програмного забезпечення у виробництві необхідне тестування локалізації, щоб переконатися, що інтерфейс користувача (UI), UX і функціональність програмного забезпечення точні для кожного регіону.

Переклад вмісту є наріжним каменем процесу локалізації, який виконують професійні лінгвісти, переважно носії цільової мови. Деякі використовують терміни «переклад» і «локалізація» як синоніми. Однак існує велика різниця між перекладом і локалізацією .

Переклад — це процес передачі вмісту з однієї мови на іншу та стосується лише письмового тексту. Зазвичай метою перекладу є якомога точніша передача оригінального повідомлення. Він включає в себе передачу змісту мови оригіналу в еквівалентному вираженні мовою перекладу так, щоб його могли зрозуміти носії цієї мови. Переклад може відбуватися в різних сферах, включаючи літературу, бізнес, юриспруденцію, техніку, медицину тощо.

Перекладати можуть як перекладачі, так і засоби машинного перекладу, наприклад, комп'ютерні програми, які використовують алгоритми для автоматичного створення перекладів. Перекладачі покладаються на свої лінгвістичні знання, знання культури та розуміння контексту, щоб точно передати зміст вихідного тексту мовою перекладу. Щоб переконатися, що перекладений матеріал є правильним, природно звучить і культурно прийнятним, вони можуть використовувати різні перекладацькі підходи, включаючи дослівний переклад, переказ, культурну адаптацію та локалізацію.

Для міжкультурних зв'язків і взаєморозуміння переклад є важливим інструментом. Він дає можливість людям з різним мовним походженням отримувати доступ до інформації, брати участь у глобальних обмінах та обмінюватися ідеями. Однак через складність мови, культурні особливості, ідіоми та контекстно-залежні значення переклад також може бути складним і вимагає ретельного вивчення для створення точного та релевантного перекладу.

Локалізація — це багаторівневий процес, який виходить за рамки простого перекладу, і включає в себе адаптацію одиниць вимірювання, стандартів часу та дати, валют, пристосування вмісту до культурних вимог, адаптацію дизайну тощо. Іноді для локалізації потрібна значна зміна вмісту актуальні для конкретного ринку. Він передбачає внесення змін до різних аспектів продукту або контенту, таких як мова, культурні особливості, форматування, дизайн і функціональність, щоб забезпечити його взаємодію з цільовою аудиторією та відповідати її очікуванням.

Локалізація часто потрібна при виході на нові ринки або обслуговуванні різноманітних аудиторій, які розмовляють різними мовами, мають різні культури та

звички. Вона передбачає глибоке розуміння регіональної культури, цінностей, звичаїв та уподобань на додаток до простого перекладу. Щоб поважати культурну чутливість і комунікаційні уподобання місцевої аудиторії, локалізація спрямована на створення безшовного користувацького досвіду, який здаватиметься їй рідним і комфортним.

Маркетингові матеріали, веб-сайти, відеоігри, програмне забезпечення, веб-сайти та інші предмети і контент можуть отримати вигоду від локалізації. Вона може передбачати зміну користувацького інтерфейсу, валюти, дати, одиниць виміру, графіки, кольорів та інших візуальних компонентів. Крім того, це може означати зміну тону, гумору, ідіом і посилань у тексті, щоб переконатися, що вони підходять для місцевої аудиторії та відповідають її культурі.

Для компаній та організацій, які прагнуть вийти на міжнародні ринки та розширити свою присутність, локалізація має важливе значення. Вона сприяє зміцненню довіри, взаємодії з клієнтами, що знаходяться поблизу, та покращує користувацький досвід загалом. Забезпечуючи зв'язок товарів чи контенту з місцевою аудиторією та задовольняючи її специфічні потреби, ефективна стратегія локалізації може збільшити частку ринку, підвищити рівень задоволеності клієнтів і сприяти зростанню бізнесу.

Незважаючи на те, що більшість інтернет-сайтів є англомовними, більшість користувачів Інтернету розмовляють не англійською мовою. Ось факт: у другому кварталі 2020 року 8 із 10 найкращих ринків [11] завантаження додатків були не англійськими (Таблиця 1.1.).

Якщо локалізацію виконати належним чином, це дозволить вашому цифровому бізнесу швидко вийти на нові ринки, почати отримувати прибуток з першого дня після випуску та випереджати конкурентів. У цьому сенсі вихід на країни з потенційно прибутковою часткою ринку, наприклад Китай [12] , може бути особливо прибутковим.

Таблиця 1.1.

Річні завантаження за країнами 2021

Країна	Завантаження (млрд)
Китай	98.3
Індія	26.6
Америка	12.1
Бразилія	10.3
Індонезія	7.3
Росія	5.5
Мексика	4.8
Туреччина	3.5
В'єтнам	3.3
Філіппіни	3.0
Пакистан	2.6
Єгипет	2.6
Японія	2.5
Таїланд	2.4
Німеччина	2.2
Великобританія	2.2

Джерело: [Data.ai]

1.2. Аналіз продуктів-конкурентів на ринку

На ринку локалізації програмних продуктів існує ряд гравців, що мають свої переваги та недоліки. Для створення власного продукту варто врахувати досягнення конкурентів та скористатися хорошими реалізованими кейсами.

Серед конкурентів: Lokalise, Phrase, Transifex, OneSky.

Lokalise (Рис. 1.1.), веб додаток який дозволяє керувати перекладами додатка, гри чи вебсайту самостійно або за допомогою команди співробітників. Створений для розробників, Lokalise пропонує такі функції:

- вбудовані пропозиції перекладу;
- проектний чат;
- вебхуки експорту та API.

Також можна протестувати та запустити переклади програми за допомогою iOS та Android.

Lokalise пропонує платні професійні послуги перекладу [7].

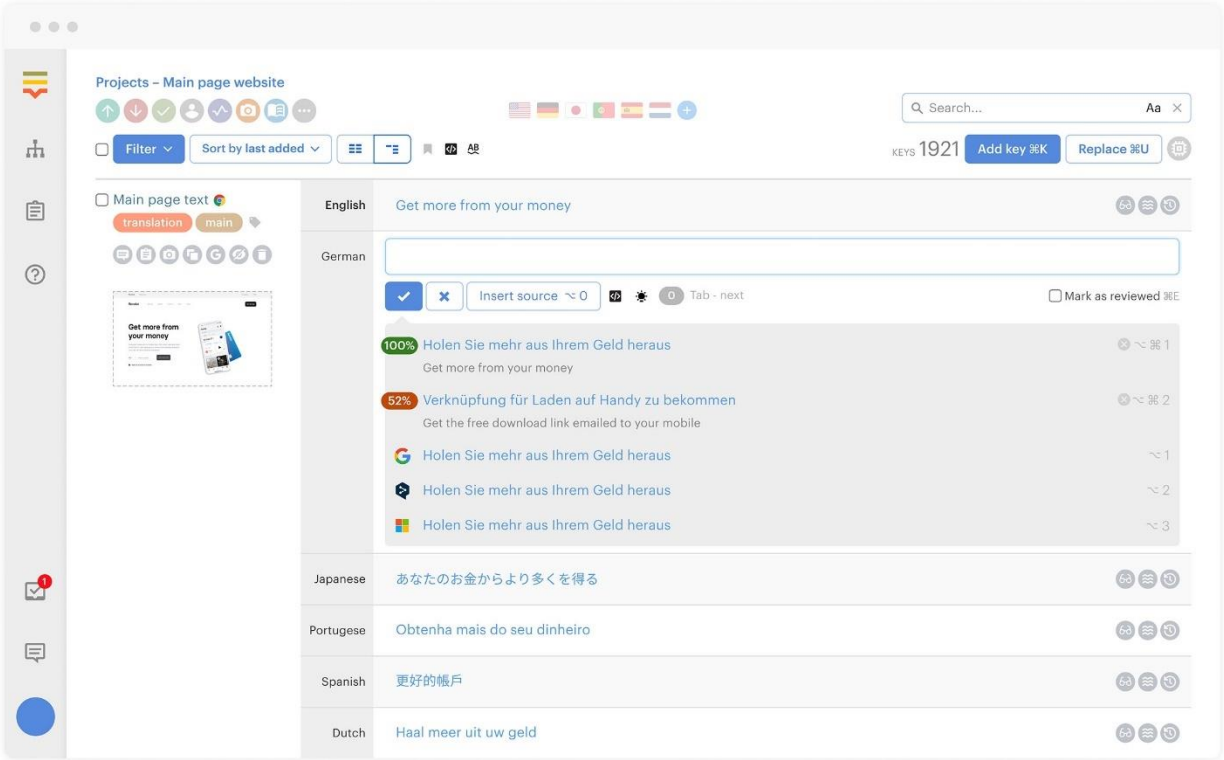


Рис. 1.1. Графічний інтерфейс програми “Lokalise”

Джерело: [https://techcrunch.com/wp-content/uploads/2020/09/Lokalise_system_light_mode.jpg]

Також, Lokalise - це платформа для управління локалізацією та перекладами, яка допомагає компаніям оптимізувати робочі процеси локалізації та керувати перекладами цифрового контенту, зокрема веб-сайтів, мобільних додатків, програмного забезпечення тощо.

Як конкурент на ринку локалізації, Lokalise має низку переваг і недоліків порівняно з іншими платформами для управління локалізацією. Детальний аналіз:

Переваги:

- Потужний функціонал для локалізації: Lokalise - це комплексне рішення для керування робочими процесами локалізації, оскільки воно надає широкий спектр функцій локалізації, як-от керування перекладами, інструменти автоматизації, забезпечення якості та можливості для співпраці.
- Зручний дизайн: Lokalise має простий дизайн, який полегшує користувачам вивчення та використання платформи. Наочна інформаційна панель дає чіткий огляд поточних проектів локалізації, їхнього прогресу та статусу, що дає змогу ефективно керувати проектами.
- Співпраця та управління командою: Завдяки таким функціям, як контроль доступу на основі ролей, пам'ять перекладів і автоматизовані процеси, Lokalise забезпечує безперешкодну співпрацю між членами команди. Це дає змогу командам ефективно співпрацювати в режимі реального часу та прискорює процес перекладу.
- Різноманітні інтеграції: Інтегрувати робочі процеси локалізації в поточні процеси розробки легко завдяки інтеграції Lokalise з відомими інструментами управління проектами, системами контролю версій і системами управління контентом. Це дає змогу ефективно керувати локалізацією, використовуючи сучасні робочі процеси та технології.

- Налаштовуваність: Lokalise дає користувачам можливість налаштувати процес локалізації відповідно до їхніх унікальних потреб. Ці можливості включають кастомні процеси, теги та атрибути. Завдяки своїй універсальності Lokalise підходить для підприємств зі специфічними потребами в локалізації.

Недоліки:

- Ціна: Для невеликих фірм або стартапів ціни на Lokalise можуть бути дуже високими, особливо якщо вони мають обмежені потреби або бюджет на локалізацію. Компанії з невеликими обсягами локалізації або нечастими потребами в локалізації можуть вважати цінові плани економічно невигідним Крива навчання: Незважаючи на зручний інтерфейс, Lokalise може потребувати певного звикання для тих, хто не знайомий з платформами керування перекладами або не має достатніх практичних знань про роботу з локалізацією. Щоб користувачі навчилися працювати з платформою, може знадобитися деякий час на навчання та адаптацію.
- Погане навчання: Незважаючи на зручний інтерфейс, Lokalise може потребувати певного звикання для тих, хто не знайомий з платформами керування перекладами або не має достатніх практичних знань про роботу з локалізацією. Щоб користувачі навчилися працювати з платформою, може знадобитися деякий час на навчання та адаптацію.
- Наразі Lokalise підтримує лише кілька постачальників послуг машинного перекладу, і деякі з них можуть не підходити для всіх користувачів або мов. Якщо користувачам потрібні переклади на мови, які не підтримуються вбудованим машинним перекладом Lokalise, можливо, їм доведеться покладатися на інші інструменти машинного перекладу.
- Обмежені альтернативи для підтримки клієнтів: Хоча Lokalise пропонує підтримку електронною поштою та чатом, вона не пропонує підтримку телефоном, що може бути недоліком для певних користувачів, які надають перевагу телефонній підтримці або потребують більш термінової допомоги.

Отже, завдяки привабливому користувацькому інтерфейсу, інструментам для спільної роботи, Lokalise надає потужне рішення для управління локалізацією. Серед його недоліків - ціна, погане навчання роботи з додатком, вибір машинного перекладу, вимоги до підключення до Інтернету та варіанти підтримки клієнтів. Обираючи Lokalise або будь-яке інше програмне забезпечення для управління локалізацією, компанії повинні ретельно оцінити свої унікальні вимоги до локалізації та бюджет.

Phrase (Рис. 1.2.) – це платформа для керування перекладами програмного забезпечення для будь-якого програмного забезпечення, наприклад мобільних або вебдодатків. За допомогою PhraseApp можна:

- керувати файлами локалізації в Інтернеті;
- працювати над проектами локалізації програмного забезпечення разом зі своєю командою;
- впроваджувати інтелектуальний робочий процес локалізації;
- конвертувати формати файлів локалізації та додавати контекстну інформацію до перекладів.

Також присутнє замовлення професійних перекладів [7].

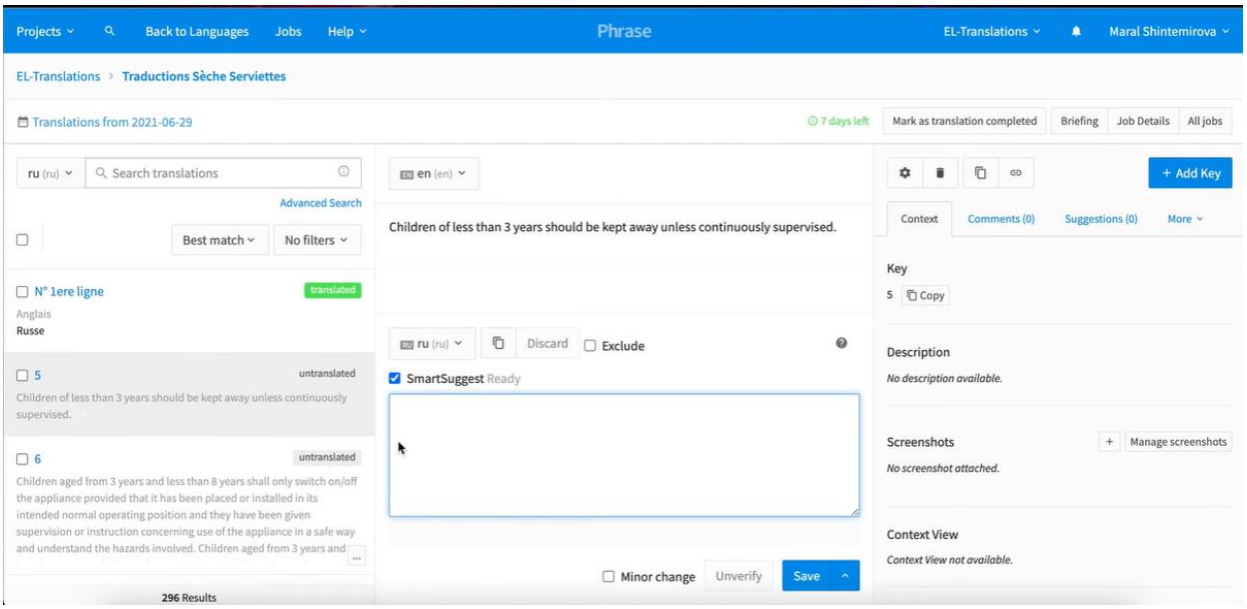


Рис. 1.2. Графічний інтерфейс програми “Lokalise”

Джерело: [https://youtu.be/ymJjqg2F9CQ]

Переваги Phrase на ринку локалізації полягають у наступному:

Платформа: Phrase надає зручну для користувача платформу з інтуїтивно зрозумілим інтерфейсом, що дозволяє користувачам легко керувати проектами локалізації та оптимізувати їх. Платформа пропонує широкий спектр можливостей, які можуть прискорити процес локалізації, зокрема керування перекладами, інструменти для співпраці та інтеграцію інструментів.

Робочі процеси, які можна налаштувати: Phrase дозволяє користувачам створювати робочі процеси, які можна налаштовувати відповідно до їхніх унікальних вимог до локалізації. Користувачі можуть створювати власні робочі процеси перекладу, встановлювати цикли перевірки та автоматизувати процеси, що підвищує продуктивність і гарантує високу якість у всіх проектах локалізації.

Комплексні функції пам'яті перекладів (TM) і керування глосарієм, які пропонує Phrase, дають змогу користувачам використовувати раніше перекладену інформацію та зберігати узгодженість перекладу. За допомогою керування пам'яттю перекладів і глосарієм можна заощадити час і гроші, скоротити витрати та підвищити якість перекладу.

Phrase надає членам команди, перекладачам і зацікавленим сторонам інструменти для співпраці та комунікації, щоб забезпечити безперешкодну комунікацію та взаємодію. Під час процесу локалізації такі функції, як контекстний перегляд перекладу, коментарі та сповіщення, допомагають покращити комунікацію, співпрацю та зворотний зв'язок.

Недоліки Phrase на ринку локалізації полягають у наступному:

- Ціна: Деяким клієнтам Phrase може здатися дорожчим за інші інструменти локалізації, доступні на ринку, що може бути недоліком для підприємств з обмеженим бюджетом або невеликих фірм.

- Хоча Phrase підтримує велику кількість мов, деякі мови або мовні пари можуть не підтримуватися, що може бути недоліком для компаній з особливими потребами в локалізації.
- Phrase використовує технологію машинного перекладу, яка не завжди забезпечує таку ж якість, як переклад, виконаний людиною, особливо для складних або спеціалізованих текстів. Для компаній, яким потрібні першокласні переклади, це може бути недоліком.
- Конкуренція: На ринку локалізації існує жорстка конкуренція, оскільки існує багато конкуруючих інструментів локалізації та компаній, що надають схожі можливості та послуги. Конкуренція, з якою стикається Phrase з боку інших відомих гравців ринку, може вплинути на її частку на ринку та перспективи зростання.

Отже, Phrase - це зручна платформа з налаштовуваними робочими процесами, надійною пам'яттю перекладів і керуванням глосарієм, а також інструментами для спільної роботи, що робить її сильним конкурентом на ринку локалізації. Однак до потенційних недоліків можна віднести ціну, обмежену мовну підтримку, якість машинного перекладу та конкуренцію з іншими інструментами локалізації. Організації, які розглядають Phrase як рішення для локалізації, повинні ретельно оцінити свої конкретні потреби, бюджет і вимоги, а також розглянути інші альтернативи на ринку, щоб прийняти обґрунтоване рішення. Ретельне дослідження, тестування та порівняння різних інструментів локалізації на основі їхніх функцій, цін і відгуків клієнтів може допомогти організаціям вибрати найбільш підходящий варіант для їхніх потреб у локалізації.

Програмний продукт Transifex (Рис. 1.3.) характеризується потужним набором інструментів і функцій перекладу, що дозволяє легко локалізувати програми та вміст від початку до кінця [7].

- зберігає весь вихідний вміст і переклади в одному місці;
- можна організувати вміст у проекти та більше ніколи не відстежувати файли вручну.

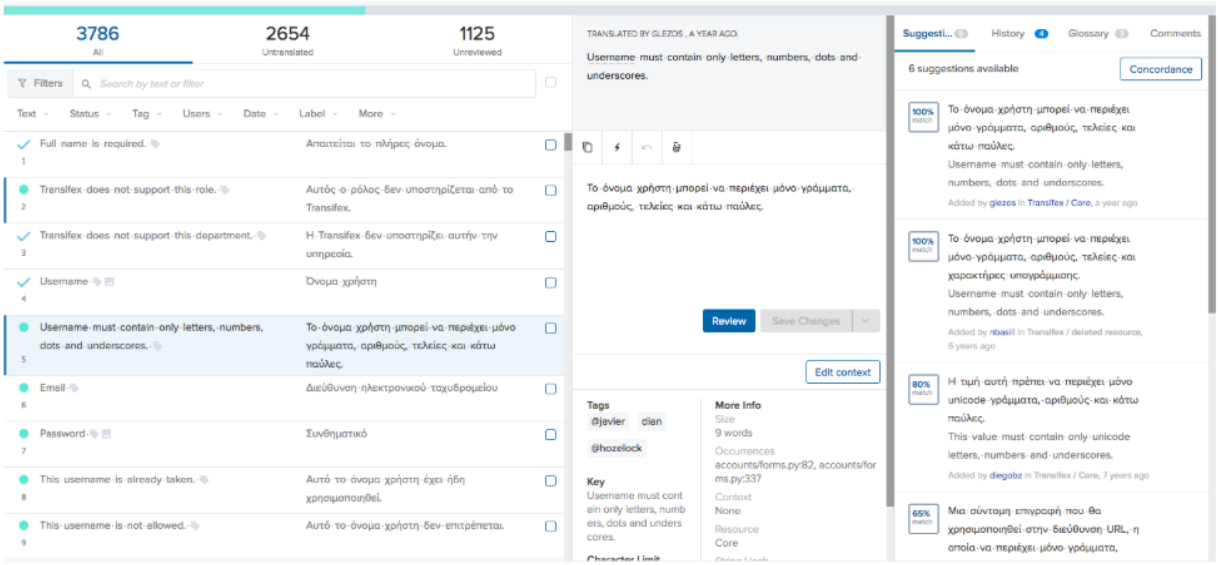


Рис. 1.3. Графічний інтерфейс програми “Lokalise”

Джерело: [https://www.transifex.com/wp-content/uploads/2019/04/drupal-transifex-integration_3.png]

Також, Transifex - це хмарна платформа локалізації, яка пропонує компаніям рішення для управління перекладами, що дозволяють перекладати їхнє програмне забезпечення, веб-сайти та документи кількома мовами. Як конкурент на ринку локалізації, Transifex має як переваги, так і недоліки.

Переваги Transifex:

- Надійна система керування перекладами: Transifex - це комплексна система керування перекладами, яка дає змогу компаніям ефективно керувати робочими процесами, співпрацювати з перекладачами та відстежувати хід перекладу. Веб-інтерфейс системи є зручним у користуванні, що дозволяє компаніям легко завантажувати вихідний контент і керувати ним, створювати перекладацькі проекти та відстежувати переклади в режимі реального часу.
- Широкий спектр підтримуваних типів файлів: Файли програмного забезпечення, документів і веб-сайтів - це лише деякі з багатьох типів файлів, які може локалізувати Transifex. Це робить його придатним для цілого ряду

підприємств і типів контенту, що робить його гнучким рішенням для організацій з різними вимогами до перекладу.

- Розширені можливості для спільної роботи: Transifex надає розширені можливості для спільної роботи, які полегшують комунікацію та злагоджену співпрацю між менеджерами проектів, перекладачами та іншими зацікавленими сторонами. Редактор перекладів спрощує забезпечення правильного перекладу, який відповідає контексту, дозволяючи перекладачам працювати над перекладом у контексті. Для прискорення процесу перекладу Transifex також пропонує систему коментування, пам'ять перекладів та інструменти управління термінологією.
- Transifex має можливість інтегруватися з відомими системами управління контентом (CMS) і системами контролю версій, такими як GitHub, GitLab і WordPress. Компанії, які вже використовують ці системи, знайдуть її корисною, оскільки вона дозволяє легко інтегрувати робочі процеси локалізації в існуючі процедури розробки та управління контентом.

Переваги Transifex:

- Вартість: Порівняно з деякими іншими платформами локалізації на ринку, Transifex може бути дещо дорожчим. Її ціноутворення базується на таких змінних, як кількість перекладацьких проектів, обсяг перекладу та кількість користувачів, що може швидко зрости для компаній із великими потребами в перекладах або великою кількістю членів команди.
- Обмежені можливості керування постачальниками: У Transifex немає ні вбудованого ринку постачальників, ні рейтингової системи для перекладачів. Через трудомісткість цього процесу компанії можуть бути змушені покладатися на сторонні програми управління постачальниками або особисто відбирати та контролювати своїх перекладачів.
- Погане навчання: Незважаючи на зручний дизайн Transifex, новим користувачам, особливо тим, хто не знайомий із процедурами локалізації або системами керування перекладами, може знадобитися деякий час, щоб

звикнути до нього. Командам, які вперше використовують платформи для локалізації, може знадобитися додаткове навчання або ввідний інструктаж.

- Хоча Transifex підтримує велику кількість типів файлів, деякі з них можуть не підтримуватися, що може обмежити його застосування для певних галузей або типів контенту. Наприклад, Transifex може не забезпечувати повну підтримку мультимедійних файлів, таких як аудіо чи відео, що може стати перешкодою для компаній, які потребують локалізації мультимедійного контенту.

Отже, як надійна хмарна платформа для локалізації, Transifex надає організаціям передові рішення для керування перекладами. Надійна система керування перекладами, підтримка різноманітних типів файлів, сучасні інструменти для спільної роботи та можливість інтеграції з популярними CMS і системами керування версіями - ось лише деякі з її переваг. Серед недоліків - тривале навчання для користувачів-початківців, цінова політика, обмежені можливості керування постачальниками й обмежена підтримка певних типів файлів. Розглядаючи Transifex як конкурента в індустрії локалізації, компаніям слід ретельно оцінити свої унікальні вимоги до локалізації, бюджет і поточні робочі процеси. Це допоможе їм вирішити, чи є Transifex найкращим варіантом для їхніх потреб.

Програмний продукт OneSky (Рис. 1.4.) пропонує комплексні рішення локалізації для будь-якого бізнесу [7].

- це лідер у галузі технічної локалізації додатків, який має досвід роботи з різноманітним контентом;
- їхнім рішенням із локалізації довіряють понад 1000 програм у понад 70 країнах.
- дозволяє автоматизувати процес локалізації;
- легко інтегрується з найпопулярнішими платформами, щоб забезпечити автоматизований процес локалізації, що скорочує час і витрати.

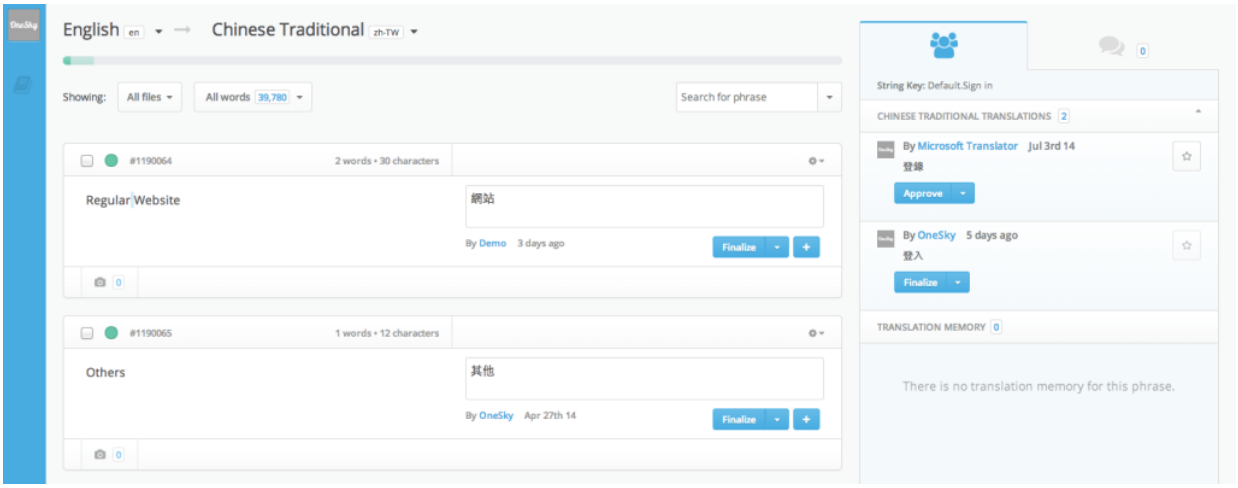


Рис. 1.4. Графічний інтерфейс програми “Lokalise”

Джерело: [https://www.oneskyapp.com/wp-content/uploads/2014/08/image02-1024x398.png]

OneSky - це хмарна платформа локалізації, яка пропонує послуги перекладу та локалізації веб-сайтів, мобільних додатків, програмного забезпечення та ігор. Як конкурент на ринку локалізації, OneSky має низку переваг і недоліків.

Переваги OneSky:

- Комплексні послуги з локалізації: OneSky пропонує широкий спектр послуг з локалізації, зокрема переклад, коректуру та тестування локалізації, що робить його універсальним рішенням для компаній, які хочуть локалізувати свій контент. Вони підтримують різні формати файлів і забезпечують інтеграцію з популярними інструментами управління проектами, роблячи процес локалізації безперебійним і ефективним.
- Переклад на основі краудсорсингу: OneSky використовує парадигму краудсорсингового перекладу, яка дозволяє компаніям залучати велику мережу перекладачів для перекладу свого контенту. Особливо для великих проектів це забезпечує швидку реакцію та доступну локалізацію.
- OneSky пропонує платформу для спільного перекладу, яка забезпечує миттєвий зворотний зв'язок і спілкування між менеджерами проектів, перекладачами та

рецензентами. Це спрощує комунікацію, забезпечує узгодженість і прискорює процес локалізації.

- OneSky також надає послуги з локалізації платформи, що передбачає переклад користувацького інтерфейсу та адаптацію функціоналу веб-сайтів, мобільних додатків і програмного забезпечення до регіональних ринків. Це корисна послуга для компаній, які бажають вийти на нові ринки та запропонувати індивідуальний клієнтський досвід.
- Потужний API та можливості інтеграції: OneSky інтегрується з відомими системами управління контентом (CMS), репозиторіями коду та іншими інструментами управління проектами, а також має власний потужний API. Це дає змогу організаціям автоматизувати процес локалізації та плавно інтегрувати його в поточні робочі процеси.

Негативні сторони OneSky:

- Контроль якості: Оскільки OneSky використовує краудсорсинговий переклад, може бути складно підтримувати однакову якість усіх перекладів. Для досягнення точних перекладів контроль якості може вимагати додаткової роботи, наприклад, ретельної перевірки та редагування.
- Незважаючи на розгалужену мережу перекладачів OneSky, вони не завжди можуть володіти необхідними лінгвістичними знаннями для вузькоспеціалізованих або технічних тем. Це може вплинути на якість і точність перекладів для певних галузей або сфер діяльності.
- Ціна: Структура цін OneSky, яка базується на оплаті за слово або за годину, може не підходити для всіх організацій. Залежно від складності контенту, це може призвести до різних витрат, що може бути економічно невигідним для невеликих проектів або бюджетів.
- OneSky - це хмарна платформа, тому компанії можуть турбуватися про безпеку та конфіденційність даних. Незважаючи на гарантії безпеки, які має OneSky, деякі компанії можуть надавати перевагу локальним рішенням для локалізації, щоб гарантувати повний контроль над своїми даними.

- Обмежена кастомізація: Платформа OneSky може бути недостатньо гнучкою та адаптивною, щоб задовольнити конкретні потреби кожної компанії в локалізації. Деяким компаніям можуть знадобитися більш спеціалізовані рішення, які можна налаштувати відповідно до їхніх унікальних вимог.

Отже, OneSky пропонує комплексну хмарну платформу локалізації з такими перевагами, як широкий спектр послуг локалізації, краудсорсинговий переклад, платформа для спільного перекладу, локалізація платформи, а також надійний API та можливості інтеграції. Однак слід враховувати й деякі недоліки, зокрема потенційні проблеми з контролем якості, обмеженість лінгвістичної експертизи, модель ціноутворення, проблеми з безпекою даних та обмежені можливості налаштування. Перш ніж розглядати OneSky або будь-яке інше рішення для локалізації як конкурента на ринку, компаніям слід ретельно оцінити свої конкретні потреби в локалізації та врахувати такі фактори, як складність контенту, бюджет, вимоги до безпеки даних і потреби в кастомізації.

Аналіз конкурентів показав, що на ринку додатків ринку присутня значна конкуренція. Конкуренти сформували потужний функціонал, але це не заважає перейняти необхідні кейси і покращити або змінити їх. Нище на основі висновків та досліджень сформований SWOT-аналіз програмного продукту “ Localiz ” (Таблиця 1.2.).

Таблиця 1.2.

SWOT-аналіз програмного продукту “ Localiz ”

Сильні сторони	Слабкі сторони
• Простий інтерфейс • Присутня історія змін • Робота над проектом в команді	• Низьке знання на ринку • Конкуренти більш розвинені в даній сфері • Відсутність достатньої бази даних для самостійного

	перекладу (залежність від сторонніх API)
Можливості	Загрози
• Залучення стороннього API для пришвидшення перекладу • Автоматичний переклад	• Зміна законодавства щодо IT сфери • Несправність орендованого сервера(сайт буде недоступним) • Поява нових конкурентів

Джерело: [створено автором]

Отже, ми проаналізували наших конкурентів і сформували SWOT-аналіз. Нашому додатку потрібно розробити всі потрібні функції, зробити дружній і зручний інтерфейс, застосувати сторонній API. API для перекладання може підвищити швидкість локалізації і покращити UX.

1.3 Постановка задачі

В результаті аналізу конкурентів було прийнято рішення перейняти декі механіки та підходи. Це сформувало бачення того як має виглядати проєкт “Localiz”.

Особливості додатку “Localiz”:

- Сторінка для гостей з короткою інформацією про додаток, яка дозволяє швидко ознайомитися з його основними перевагами.
- Простий інтерфейс, що спрощує використання додатку і забезпечує зручність управління.
- Історія перекладу файлів, яка зберігає всі зміни, зроблені в процесі перекладу, для зручного відстеження та відновлення попередніх версій.

- Оптимізація та пагінація даних, що дозволяє ефективно обробляти та відображати великі обсяги інформації та поділити її на сторінки для зручного перегляду.
- Безкоштовне користування додатком, що дає можливість користувачам використовувати всі функції та можливості без необхідності оплати.

Додаток матиме наступні функції та можливості як:

- Сторінка для гостей з короткою інформацією про додаток, яка дозволяє швидко ознайомитися з його основними перевагами.
- Простий інтерфейс, що спрощує використання додатку і забезпечує зручність управління.
- Історія перекладу файлів, яка зберігає всі зміни, зроблені в процесі перекладу, для зручного відстеження та відновлення попередніх версій.
- Оптимізація та пагінація даних, що дозволяє ефективно обробляти та відображати великі обсяги інформації та поділити її на сторінки для зручного перегляду.
- Безкоштовне користування додатком, що дає можливість користувачам використовувати всі функції та можливості без необхідності оплати.

Отже, ми описали особливості додатку “Localiz”, також його функції та можливості.

1.4 Організація роботи

“Localiz” розробляється однією людиною і для того щоб пришвидшити і полегшити роботу, також надати можливість іншим людям переглядати результати роботи, було вирішено розмітити проект на GitHub.

GitHub - це веб-платформа, яка надає централізоване місце для контролю версій та співпраці над програмними проектами. Вона дозволяє розробникам розміщувати свої сховища коду, відстежувати зміни в своїй кодовій базі та співпрацювати з іншими над завданнями розробки.

GitHub пропонує різні функції та інструменти, зокрема:

- Сховища коду: GitHub дозволить створювати та розміщувати репозиторії Git, що полегшує управління та відстеження змін у їхній кодовій базі.
- Розгалуження та злиття: Надає можливість створювати гілки для роботи над новими функціями або експериментів, не впливаючи на основну кодову базу. Потім можна об'єднати ці гілки назад в основну гілку.
- Pull-запити: За допомогою pull-запитів можна пропонувати зміни до кодової бази і просити інших переглянути та об'єднати їхній код. Це полегшує співпрацю та перегляд коду між членами команди.
- Відстеження проблем: GitHub надає систему відстеження проблем, що дозволяє користувачам створювати, призначати та відстежувати проблеми або помилки в проекті. Це допомагає в організації та визначенні пріоритетів завдань.
- Вікі та сторінки проектів: GitHub дозволяє створювати вікі та сторінки проекту для документування та обміну інформацією, пов'язаною з проектом.
- Інтеграція з іншими інструментами: GitHub інтегрується з різними інструментами та сервісами для розробки, такими як системи безперервної інтеграції (CI), редактори коду, інструменти управління проектами тощо.

GitHub став популярною платформою серед розробників та проектів з відкритим вихідним кодом завдяки зручному інтерфейсу, потужним функціям співпраці та широкій підтримці спільноти.

Отже, так як проєкт розроблятиметься однією людиною для організації розробки додатку використовуватиметься тільки GitHub.

Висновки до розділу 1

У розділі 1 було проаналізовано значення локалізації програмних додатків у сучасному глобалізованому світі. Враховуючи ключову роль перекладу контенту в процесі локалізації, необхідно було створити зручний додаток для перекладу, який би

спростив робочий процес перекладу. Таким чином, було прийнято рішення розробити додаток для перекладу під назвою “Localiz”.

Щоб забезпечити конкурентоспроможність “Localiz” на ринку, було всебічно вивченно галузь локалізації додатків і виявилино численних конкурентів із потужним функціоналом. Відповідно, було проведено ретельне дослідження для аналізу їхніх особливостей, що врешті-решт призвело до включення окремих елементів конкуретів до вебдодатка “Localiz”. Цей стратегічний крок мав на меті розширити функціональність додатка “Localiz” та зміцнити його позиції на ринку.

Спираючись на результати аналізу конкурентів, було сформульовано чіткий план роботи. План охоплював різні аспекти, зокрема створення гостьової сторінки, яка надавала б стислу інформацію про програму, реалізацію зручного інтерфейсу, включення функції історії перекладу файлів, оптимізацію обробки даних і пагінації, а також, що особливо важливо, можливості атоматичного перекладу.

Впроваджуючи ці функції, “Localiz” мав на меті забезпечити безперебійну та ефективну роботу користувачів, а також задовольнити різноманітні потреби та вподобання своїх користувачів. Такий комплексний підхід забезпечив для застосунку “Localiz” місце провідного гравця на ринку локалізації програмних додатків, здатного задовольнити потреби все більш глобалізованого світу.

РОЗДІЛ 2
РОЗРОБКА ВЕБЗАСТОСУНКУ “LOCALIZ”

2.1. Опис технологічного стеку та архітектури рішення

«Якщо ви думаєте, що хороша архітектура коштує дорого, спробуйте погану архітектуру». - Браян Фут і Джозеф Йодер [13].

Архітектура вебдодатків — це схема одночасної взаємодії між компонентами, базами даних, системами проміжного програмного забезпечення, інтерфейсами користувача та серверами в додатку. Його також можна описати як макет, який логічно визначає зв’язок між сервером і стороною клієнта для кращої взаємодії та роботи в Інтернеті.

Ефективна, якісна вебархітектура стала де-факто хорошим рішенням для створення програмного продукта і дійсно є інструментом для формування чіткого потоку даних та інформації щоб досягнути бажаних бізнес-цілей. Архітектура з продуманими функціями та інтуїтивно зрозумілим інтерфейсом забезпечує безперебійну роботу користувача. Це також зменшує ймовірність збою програми і легке його вирішення.

Так як мета проекту передбачає створення не просто вебсайта, а послуг повноцінного вебдодатку є доцільним використовувати саме багаторівневу архітектуру проекту “N-Layer”. Найбільш поширена організація логіки додатків побудована на рівнях показана на Рис. 2.1..

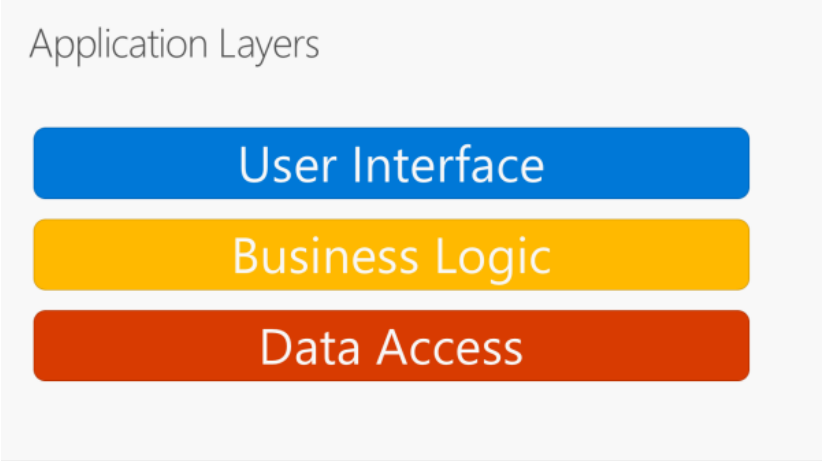


Рис. 2.1. Типові шари в “N-Layer” архітектурі

Джерело: [13]

Ці рівні часто називають такими абревіатурами як UI, BLL (Business Logic Layer) і DAL (Data Access Layer). Використовуючи цю архітектуру, користувачі роблять запити через рівень інтерфейсу користувача, який взаємодіє лише з BLL. BLL, у свою чергу, може викликати DAL для запитів доступу до даних. Рівень інтерфейсу користувача не повинен надсилати жодних запитів напряму до DAL, а також не повинен взаємодіяти з рівень представлення даних (Persistence) безпосередньо за допомогою інших засобів. Відповідним чином, BLL має взаємодіяти лише з рівень представлення даних, проходячи через DAL. Таким чином, кожен рівень має свою власну добре відому відповідальність [13].

2.1.1. Опис архітектури бекенду вебзастосунку “Localiz”

“N-Layer” архітектура розвивалася і поклала початок новим рішенням. Програми, які дотримуються принципу інверсії залежностей, а також принципів Предметно-орієнтоване проєктування (DDD або Domain-Driven Design), як правило, мають схожу архітектуру. Протягом багатьох років ця архітектура мала багато імен. Однією з перших назв була Hexagonal Architecture, а потім Ports-and-Adapters. Нещодавно її називали цибулевою архітектурою (Onion Architecture) або чистою архітектурою (Clean Architecture). Остання назва, чиста архітектура (Рис. 2.1.), використовується до сьогодні. [13]

Це все підвило до використання в проекті саме Чистої архітектури. Розробка бекенду відбувалась на фрейворку ASP.NET Core 6. Більшість традиційних програм .NET розгортаються як окремі блоки, що відповідають виконуваному файлу або одній вебпрограмі, що працює в одному домені програми IIS. Цей підхід є найпростішою моделлю розгортання і дуже добре обслуговує багато внутрішніх і менших загальнодоступних програм. Однак, навіть враховуючи цю єдину одиницю розгортання, більшість нетривіальних бізнес-додатків виграють від деякого логічного поділу на кілька рівнів.

ASP.NET Core 6 побудований на мові програмування C#, а вона у свою чергу є ООП орієнтованою і дозволить легко реалізувати вибране архітектурне рішення.

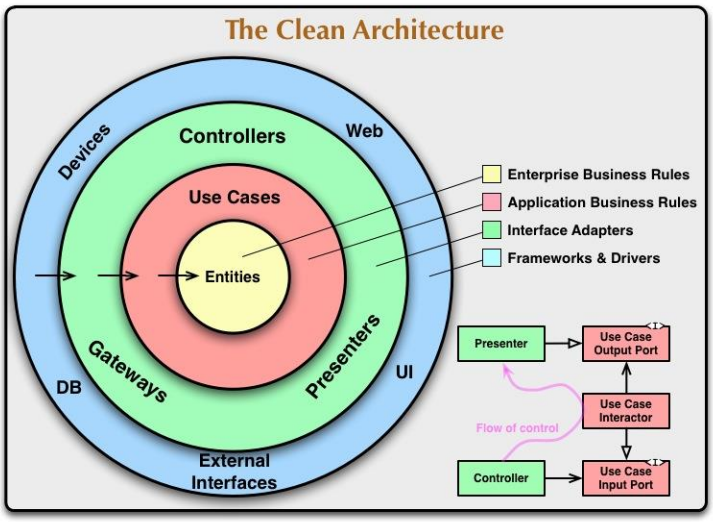


Рис. 2.2. Clean Architecture Diagram

Джерело: [8]

Будова проекта складається з (Рис. 2.3.) головного LocalizAPI проекту та 2-х бібліотек, а саме Core який містить всю бізнес логіку, та таблиці і моделі для взаємодії з БД та Infrastructure яка відповідає за підключення і налаштування БД, розміщення міграції.

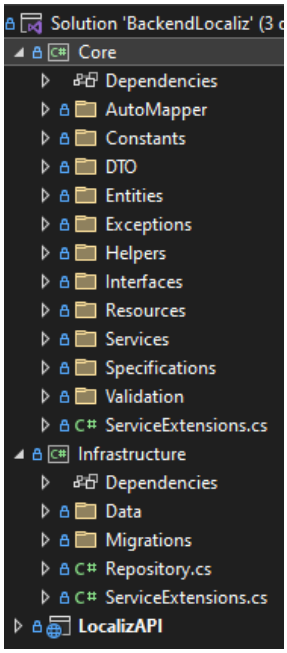


Рис. 2.3. Структура рішення серверної частини програмного продукту “localiz”

Джерело: [створено автором]

2.1.1.1. Стек технологій використаних у розробці бекенду вебзастосунку “Localiz”

ASP.NET Core 6 - це передове, багатофункціональне, кросплатформенне, високопродуктивне середовище з відкритим вихідним кодом, спеціально розроблене для розробників, які створюють і розгортають сучасні хмарні додатки, підключені до Інтернету. Завдяки своїм потужним можливостям і гнучкості, ASP.NET Core 6 дозволяє розробникам створювати масштабовані, надійні та безпечні рішення, які відповідають вимогам сучасного цифрового ландшафту, що швидко розвивається.

Будучи крос-платформною платформою, ASP.NET Core 6 надає розробникам свободу у створенні додатків, які можуть безперешкодно працювати на різних операційних системах, включаючи Windows, macOS і Linux. Ця гнучкість дозволяє розробникам використовувати наявні навички та інструменти, орієнтуючись на широкий спектр платформ, забезпечуючи максимальне охоплення та доступність своїх додатків.

Забезпечення виняткової продуктивності лежить в основі ASP.NET Core 6. Завдяки різноманітним оптимізаціям та покращенням продуктивності розробники

можуть досягти блискавичного часу відгуку, зменшення затримок та ефективного використання ресурсів. Ці вдосконалення дозволяють додаткам, створеним на основі ASP.NET Core 6, справлятися з високими навантаженнями і забезпечувати винятковий користувацький досвід навіть в умовах інтенсивного трафіку.

Будучи фреймворком з відкритим вихідним кодом, ASP.NET Core 6 сприяє створенню динамічної спільноти розробників, які беруть участь у його постійному розвитку та вдосконаленні. Ця відкритість сприяє інноваціям і дозволяє розробникам використовувати колективні знання та досвід спільноти, забезпечуючи відповідність фреймворку останнім галузевим стандартам і передовим практикам.

ASP.NET Core 6 охоплює сучасні парадигми та патерни розробки, що робить його ідеальним вибором для створення хмарних додатків. Він легко інтегрується з популярними хмарними платформами, такими як Azure і AWS, і забезпечує готову підтримку технологій контейнеризації, таких як Docker і Kubernetes. Ця вбудована інтеграція спрощує процес розгортання та масштабування, дозволяючи розробникам використовувати весь потенціал хмари та створювати високомасштабовані та відмовостійкі додатки.

Безпека є головним пріоритетом в ASP.NET Core 6. Фреймворк пропонує надійні функції та механізми безпеки для захисту додатків від поширених веб-уразливостей, таких як міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF). Крім того, ASP.NET Core 6 надає вбудовану підтримку автентифікації та авторизації, гарантуючи, що тільки авторизовані користувачі можуть отримати доступ до конфіденційних ресурсів і виконувати привілейовані дії [3].

Web API - це потужний і універсальний фреймворк для побудови надійних HTTP-сервісів, доступних з широкого кола клієнтів, таких як веб-браузери, мобільні пристрої і навіть інші сервери. Завдяки своїй гнучкості та сумісності, Web API слугує чудовою платформою для розробки RESTful-додатків на платформі .NET Framework.

Використовуючи Web API, розробники можуть проектувати і впроваджувати API, які дотримуються принципів передачі представницького стану (REST),

забезпечуючи стандартизований і ефективний підхід до проектування веб-сервісів. Ці API забезпечують безперебійний зв'язок і обмін даними між клієнтами та серверами, полегшуючи створення масштабованих і функціонально сумісних додатків.

Однією з ключових переваг Web API є його здатність підтримувати різні формати даних, включаючи JSON (JavaScript Object Notation) і XML (eXtensible Markup Language). Така гнучкість дозволяє розробникам обирати формат, який найкраще відповідає потребам їхнього додатку, забезпечуючи сумісність з різними клієнтськими технологіями.

Крім того, Web API пропонує широкий спектр можливостей для покращення процесу розробки. Він забезпечує надійні механізми маршрутизації, які дозволяють розробникам визначати чисті та інтуїтивно зрозумілі URL-адреси для своїх API. Крім того, він підтримує узгодження вмісту, дозволяючи клієнтам запитувати певні формати даних, а серверу відповідати відповідно [14].

Бібліотека класів або **DLL** (Dynamic Link Library) - це фундаментальний компонент у розробці програмного забезпечення, який інкапсулює програмний код, дані та ресурси. Вона слугує багаторазовим модулем, який може бути використаний іншими програмами, полегшуючи обмін кодом і підвищуючи ефективність процесів розробки програмного забезпечення. Створюючи бібліотеку класів або DLL, розробники можуть консолідувати часто використовувані функції, алгоритми та структури даних в одному пакеті.

Однією з важливих переваг бібліотеки класів або DLL є її легка інтеграція з різними проектами Visual Studio. Visual Studio, широко використовуване інтегроване середовище розробки (IDE), надає повний набір інструментів і функцій для розробки програмного забезпечення. Коли бібліотека класів або DLL створена у Visual Studio, вона може бути легко інтегрована в різні проекти, що дозволяє розробникам використовувати її функціональність без необхідності переписувати або дублювати код.

Універсальність бібліотеки класів або DLL полягає в її здатності абстрагувати складні операції та пропонувати їх у вигляді простих, багаторазових компонентів. Це дозволяє розробникам інкапсулювати свій код у класи, простори імен та модулі, забезпечуючи чіткий та структурований підхід до розробки програмного забезпечення. Інкапсулюючи код у бібліотеку класів або DLL, розробники можуть сприяти повторному використанню коду, модульності та підтримці, що призводить до створення більш ефективних та масштабованих програмних рішень.

Крім того, бібліотека класів або DLL може охоплювати не лише програмний код, але й дані та ресурси. Це означає, що поряд з методами і класами вона може включати структури даних, конфігураційні файли, графічні ресурси та інші важливі компоненти, необхідні для функціонування програмного забезпечення. Консолідуючи ці елементи в межах бібліотеки класів або DLL, розробники можуть створювати автономні пакети, які легко поширювати, оновлювати та керувати ними.

Загалом, використання бібліотек класів або DLL надає значні переваги у розробці програмного забезпечення. Вони сприяють повторному використанню коду, підвищують модульність і зручність супроводу, спрощують інтеграцію з проектами Visual Studio та інкапсулюють програмний код, дані і ресурси в портативні пакети. Використовуючи можливості бібліотек класів або DLL, розробники можуть оптимізувати свої робочі процеси та створювати надійні, масштабовані та ефективні програмні рішення. [15]

Microsoft SQL Server - це надійна та масштабована система управління реляційними базами даних (СКБД), розроблена корпорацією Microsoft. Вона призначена для обробки великих обсягів даних і забезпечує ефективне зберігання, пошук і керування реляційними даними. SQL Server пропонує повний набір функцій та інструментів, які дозволяють організаціям легко створювати, розгортати та керувати своїми додатками для роботи з базами даних.

Однією з ключових переваг Microsoft SQL Server є його універсальність, оскільки він підтримує широкий спектр типів даних і надає потужні можливості маніпулювання даними. Він дозволяє користувачам визначати складні запити,

виконувати агрегації та виконувати збережені процедури для виконання різних операцій з даними. SQL Server також пропонує розширені можливості, такі як індексування, розбиття на розділи та повнотекстовий пошук, які підвищують продуктивність запитів та забезпечують ефективний пошук даних.

Крім того, SQL Server забезпечує цілісність і безпеку даних завдяки надійним можливостям транзакцій і вбудованим механізмам захисту. Він підтримує властивості ACID (атомарність, узгодженість, ізоляція, довговічність), гарантуючи, що транзакції обробляються надійно і послідовно. SQL Server також надає різні функції безпеки, такі як автентифікація, авторизація та шифрування для збереження конфіденційних даних і захисту від несанкціонованого доступу.

На додаток до основних функцій баз даних, Microsoft SQL Server легко інтегрується з іншими продуктами та технологіями Microsoft. Він має чудову сумісність з платформою .NET, що дозволяє розробникам створювати потужні та масштабовані додатки, використовуючи знайомі мови програмування, такі як C# та VB.NET. SQL Server також забезпечує інтеграцію з Microsoft Azure, що дозволяє безперешкодно мігрувати та розгортати бази даних у хмарі, а також використовувати хмарні сервіси для забезпечення масштабованості та високої доступності [16].

Entity Framework Core - це універсальна, багатофункціональна та незалежна від платформи ітерація відомої технології доступу до даних Entity Framework з відкритим кодом. Вона дозволяє розробникам без особливих зусиль створювати комплексні моделі баз даних, що відповідають різним типам баз даних, незалежно від операційної системи. Завдяки своїй легкій конструкції та широким можливостям розширення, Entity Framework Core забезпечує ефективний та гнучкий підхід до управління збереженням даних у сучасних програмних додатках.

Завдяки легкій інтеграції з багатьма постачальниками баз даних, він забезпечує безперешкодну взаємодію з різними системами баз даних, включаючи реляційні бази даних, бази даних NoSQL і хмарні рішення для зберігання даних. Незалежно від того, чи працюєте ви над десктопним додатком, вебпроєктом або навіть мобільним додатком, Entity Framework Core слугує надійним і масштабованим фреймворком для

спрощення операцій з базами даних, полегшення ефективного запиту даних і забезпечення безперебійної синхронізації між вашим додатком і основним джерелом даних.

Завдяки широкому набору функцій та вичерпній документації Entity Framework Core надає розробникам інструменти, необхідні для створення надійних і продуктивних рівнів доступу до даних, підвищуючи продуктивність і забезпечуючи швидку розробку додатків, керованих базами даних [3].

Ardalis.Specification.EntityFrameworkCore – пакет специфікацій надає базовий клас для того, щоб розробники почали створювати специфікації в контексті Domain-Driven Design. Пакет також містить інтерфейси для створення сховищ, сумісних зі специфікаціями, а також інтерфейси, які налаштовують поведінку специфікацій [9].

AutoMapper - це потужний та ефективний пакет, призначений для спрощення процесу проектування однієї моделі на іншу, що призводить до значного зменшення складності коду та підвищення простоти програми. Використовуючи AutoMapper, стає можливим легко трансформувати об'єкти з одного представлення в інше, заощаджуючи цінний час і зусилля в процесі розробки.

Цей універсальний пакет ідеально підходить для сценаріїв, коли потрібно зіставити властивості одного об'єкта з відповідними властивостями іншого об'єкта, навіть якщо ці дві моделі можуть відрізнятися за структурою або іменуванням. Визначаючи конфігурації, AutoMapper інтелектуально обробляє перетворення даних, гарантуючи, що цільовий об'єкт точно відображає бажаний стан.

Переваги використання AutoMapper виходять за рамки скорочення коду і спрощення програми. Завдяки інтуїтивно зрозумілому синтаксису та простим у використанні функціям, ми можемо зосередитися на основній логіці свого додатку замість того, щоб писати повторюваний і схильний до помилок код. Це підвищує продуктивності та зручності підтримки, дозволяючи розробникам швидко адаптуватися до мінливих вимог і створювати високоякісні програмні рішення [6].

Fluent Validation - популярна бібліотека .NET для створення строго типізованих правил перевірки[7].

JwtBearer – це широко використовуване проміжне програмне забезпечення для автентифікації у веб-розробці, яке забезпечує безпечний та ефективний спосіб обробки JSON-токенів (JWT) для автентифікації та авторизації користувачів. Створений спеціально для екосистеми .NET, JwtBearer спрощує процес перевірки та обробки JWT, дозволяючи з легкістю впроваджувати надійні механізми автентифікації у додаток.

Використовуючи JwtBearer, ми можемо легко інтегрувати автентифікацію на основі JWT у свій додатк. Це проміжне програмне забезпечення перевіряє вхідний JWT, забезпечуючи його автентичність і цілісність, і витягує відповідні твердження та інформацію, необхідну для авторизації та ідентифікації користувача.

JwtBearer підтримує різні схеми автентифікації, включаючи перевірку JWT на основі симетричного ключа, асиметричного ключа або сертифіката. Така гнучкість дозволяє обирати механізм автентифікації, який найкраще відповідає вимогам безпеки їхнього додатку.

Однією з помітних переваг JwtBearer є його здатність безперешкодно обробляти закінчення терміну дії та оновлення токенів. Він автоматично перевіряє термін дії токену і надає можливість автоматичного оновлення токену без додаткового ручного втручання.

Крім того, JwtBearer легко інтегрується з іншими популярними фреймворками і технологіями автентифікації, такими як ASP.NET Core Identity і OAuth 2.0, дозволяючи розробникам використовувати існуючі системи автентифікації і розширювати їх за допомогою автентифікації на основі JWT [1].

Swagger є відкритим стандартом для визначення та документування RESTful API. Він забезпечує машинозчитуваний формат (зазвичай у JSON або YAML) для опису кінцевих точок API, корисного навантаження запиту/відповіді, методів автентифікації та інших важливих деталей. Специфікація дозволяє розробникам

створювати інтерактивну документацію API, клієнтські SDK, серверні заглушки та інші інструменти [5].

2.1.2 Опис архітектури фронтенду вебзастосунку “Localiz”

Розробка фронтенду відбувалась на (Рис. 2.4.) бібліотеці React і допоміжними бібліотеками Redux, Material UI, I18n.

Рис. 2.4. Бібліотеки для фронтенду

Джерело: [створено автором]

Вибір шаблону дизайну є свідомим рішенням, яке не повинно впливати на продуктивність програми. Тут реалізується шаблон «Container – View pattern». «Container – View pattern» - це патерн проектування, який використовується в розробці програмного забезпечення, зокрема в дизайні інтерфейсу користувача (UI), щоб відокремити логіку представлення від логіки управління даними. Він зазвичай використовується в додатках з графічним інтерфейсом користувача (GUI), таких як десктопні та мобільні додатки, щоб створити чіткий розподіл завдань і сприяти зручності супроводу та масштабованості.

Контейнер і представлення - це дві основні частини інтерфейсу користувача (UI) в парадигмі Контейнер-Представлення. Представлення відповідає за візуалізацію інтерфейсу та отримання даних від користувача, а Контейнер відповідає за управління даними та бізнес-логіку. За допомогою чітко визначених інтерфейсів або контрактів Контейнер і Представлення можуть взаємодіяти один з одним.

Логіка управління даними, включаючи отримання даних із зовнішніх джерел, обробку даних і контроль стану додатку, міститься в Контейнері. Він часто реалізує бізнес-логіку, включаючи обробку, агрегування та валідацію даних. Контейнер слугує

сполучною ланкою між Представленням та основними джерелами даних або сервісами, надаючи API або інтерфейс для Представлення, щоб воно могло взаємодіяти з даними.

Представлення (View), з іншого боку, відповідає за рендеринг користувацького інтерфейсу та надання користувачеві даних. У більшості випадків він не має бізнес-логіки і є неактивним. Вигляд коригує інтерфейс після отримання оновлень від Контейнера. Вхідні дані від користувачів також приймаються і надсилаються до Контейнера для обробки.

Патерн Контейнер-Вигляд заохочує розподіл праці і допомагає встановити чітку межу між користувацьким інтерфейсом і логікою управління даними. Якщо зміни в інтерфейсі або логіці управління даними можуть бути внесені окремо один від одного, не впливаючи один на одного, це може покращити зручність супроводу, масштабованість та тестування додатку.

Використання парадигми Container-View має кілька переваг, таких як:

1. Розділення завдань: Відокремлюючи функціональність інтерфейсу та управління даними, патерн робить програмне забезпечення простішим для розуміння та підтримки.
2. Можливість багаторазового використання: Контейнер можна використовувати з іншими представленнями, що робить дизайн більш модульним та універсальним.
3. Масштабованість: Допомагаючи керувати складними користувацькими інтерфейсами з різними поданнями та джерелами даних, патерн полегшує масштабування програми.
4. Тестування: Завдяки розділенню завдань, простіше створювати модульні тести для Контейнера та Представлення окремо, що призводить до більшого тестового покриття та надійнішого програмного забезпечення.
5. Гнучкість: Оскільки Контейнер може бути побудований незалежно від Вигляду, патерн пропонує гнучкість у використанні різних технологій інтерфейсу користувача або фреймворків.

Отже, патерн «Container – View pattern» - це патерн проектування, який підтримує розподіл обов'язків при розробці інтерфейсу користувача, розбиваючи його на дві основні частини: Контейнер і Вигляд. У той час як представлення відповідає за рендеринг інтерфейсу і введення даних користувачем, контейнер відповідає за управління даними і бізнес-логіку. За допомогою цього дизайну можна покращити зручність обслуговування, масштабованість та тестування інтерфейсу користувача.

Одним із інших відомих шаблонів дизайну є «Atomic pattern». Весь функціонал розділений по папках з відповідною назвою яка розкриває його сенс (Рис. 2.5.).

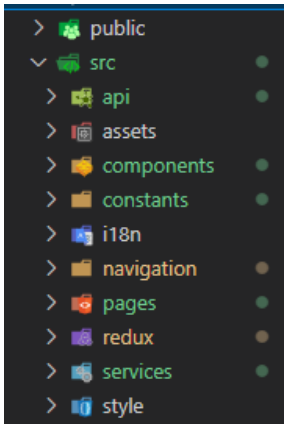


Рис. 2.5. Структура фронтенду

Джерело: [створено автором]

2.1.2.1. Стек технологій використаних у розробці фронтенду вебзастосунку “localiz”

React – це бібліотека JavaScript з відкритим вихідним кодом для створення користувацьких інтерфейсів, зокрема для вебдодатків. Вона була створена компанією Facebook і широко використовується розробниками для створення сучасних, інтерактивних та динамічних користувацьких інтерфейсів.

Компонентний дизайн, що використовується в React, дозволяє представляти елементи інтерфейсу як багаторазові компоненти. Ці елементи можна комбінувати для створення складних користувацьких інтерфейсів, оскільки кожен з них містить логіку інтерфейсу. React оптимізує продуктивність, використовуючи віртуальну

DOM (Document Object Model) для швидкого оновлення та рендерингу компонентів інтерфейсу.

Розробники можуть вказати, як повинен виглядати інтерфейс користувача (UI) на основі поточного стану, а React подбає про оновлення UI, коли стан зміниться. React пропонує такий декларативний підхід до розробки інтерфейсу користувача. React є дуже популярним варіантом для створення вебдодатків, оскільки він чудово працює з іншими JavaScript бібліотеками та фреймворками.

Основні характеристики React включають наступне:

- Елементи інтерфейсу представлені як багаторазові компоненти в компонентно-орієнтованому дизайні.
- Віртуальний DOM: Швидке оновлення та рендеринг елементів інтерфейсу користувача.
- Декларативний підхід: Визначення того, як повинен виглядати інтерфейс користувача у світлі поточної ситуації.
- Передбачуваний потік даних та оновлень у додатку забезпечується одностороннім зв'язуванням даних.
- Розробники можуть створювати код JavaScript, що нагадує HTML, використовуючи розширення синтаксису JSX.
- Дані рухаються в одному напрямку з односпрямованим потоком даних, що є простішим для розуміння та налагодження.

Односторінкові додатки (SPA) часто створюються за допомогою React, який часто використовується з іншими технологіями, такими як Redux для управління станами, React Router для маршрутизації та Axios для запитів до API, серед інших. Він має значну та енергійну спільноту розробників і постійно змінюється завдяки частим оновленням та вдосконаленням.[20]

Material-UI - це популярний фреймворк інтерфейсу користувача з відкритим вихідним кодом для створення вебдодатків за допомогою React, бібліотеки JavaScript для побудови користувацьких інтерфейсів. Material-UI надає набір заздалегідь

розроблених компонентів, що налаштовуються, які відповідають принципам Material Design - мови дизайну, розробленої Google, яка фокусується на чистому, сучасному та адаптивному дизайні.

Розробники можуть використовувати різноманітні компоненти Material-UI, включаючи кнопки, форми, діалоги, меню, вкладки, піктограми та багато іншого, щоб створювати візуально привабливі та цікаві користувацькі інтерфейси. Крім того, він надає універсальну систему тем, яка дозволяє розробникам змінювати зовнішній вигляд своїх додатків відповідно до специфікацій бренду або дизайну.

Адаптивний дизайн, підтримка доступності, підтримка інтернаціоналізації (i18n) та ретельна документація з прикладами та демонстраціями - ось деякі з основних аспектів Material-UI. Розробники з широкої та динамічної спільноти Material-UI постійно підтримують та оновлюють фреймворк.

Загалом, Material-UI - це потужний і популярний фреймворк інтерфейсу, який може полегшити створення сучасних, адаптивних онлайн-додатків за допомогою React, дотримуючись при цьому принципів Material Design. [21].

I18n – функції інтернаціоналізації нашого розширення. Ми можем використовувати ці API, щоб отримати локалізовані рядки з файлів локалізації, що містяться у вашому розширенні, дізнатися поточну мову браузера та значення його заголовка Accept-Language.

I18n - це скорочення від інтернаціоналізації, яка є практикою розробки програмного забезпечення, що передбачає проектування та розробку додатків або систем таким чином, щоб їх можна було легко адаптувати для різних мов, регіонів та культур. "I" в I18n - це перша літера "I", за якою йдуть 18 літер, а потім літера "n", яка вказує на те, що між ними пропущено 18 літер, що робить цей термін скороченим.

Адаптація програмного коду та користувацького інтерфейсу до різних мов, форматів дати й часу, числових форматів, кодування символів та інших культурних і регіональних відмінностей називається інтернаціоналізацією. Це дає змогу локалізувати або перекласти програмне забезпечення без зміни основної кодової бази

для різних цільових ринків. Програмне забезпечення, яке хоче бути доступним для користувачів з різним мовним і культурним досвідом, повинно враховувати I18n. [22]

SweetAlert2 - популярна бібліотека JavaScript для створення красивих діалогових вікон сповіщень, що налаштовуються. Це оновлена версія оригінальної бібліотеки SweetAlert з більшими можливостями та гнучкістю. SweetAlert2 дозволяє розробникам створювати красиві та адаптивні сповіщення, підказки та підтвердження з власними стилями, анімацією та кнопками.

SweetAlert2 має наступні основні характеристики:

1. Стилі, які можна змінювати: SweetAlert2 надає розробникам можливість змінювати візуальний стиль діалогів сповіщень за допомогою класів CSS, забезпечуючи уніфікований зовнішній вигляд програми.
2. SweetAlert2 підтримує як модальні, так і немодальні діалоги. Модальні діалоги не дозволяють користувачеві взаємодіяти з рештою сторінки, в той час як немодальні діалоги дозволяють користувачеві взаємодіяти зі сторінкою, поки діалог відкритий.
3. Анімаційні ефекти: Щоб покращити взаємодію з користувачем, додати естетичної привабливості та інтерактивності, SweetAlert2 пропонує набір вбудованих анімаційних ефектів для відображення та приховування діалогових вікон сповіщень.
4. Користувацькі кнопки та зворотні виклики SweetAlert2 мають настроювані мітки, стилі та зворотні виклики, які забезпечують різноманітні взаємодії та дії, засновані на введені користувачем.
5. Підтримка обіцянок: SweetAlert2 пропонує підтримку обіцянок, що дозволяє легко керувати асинхронною взаємодією користувачів та об'єднувати декілька діалогів.
6. Локалізація: Завдяки підтримці локалізації SweetAlert2, розробники можуть показувати діалоги сповіщень різними мовами залежно від місця розташування користувача.

7. Доступність: SweetAlert2 є зручним для людей з обмеженими можливостями завдяки підтримці клавіатурної навігації та зчитувачів з екрану, які були включені в процес розробки.

Отже, SweetAlert2 - це потужний та універсальний інструментарій для створення кастомізованих та зручних діалогових вікон сповіщень у JavaScript-додатках. Він активно оновлюється новими функціями та виправленнями помилок і має значну спільноту користувачів та розробників.[17]

2.1.3 Опис реалізації вебсервісу

Головним користувачем вебдодатку “Lokaliz” будуть локалізатори та перекладачі. Програма має надати користувачеві набір інструментів які дозволять легко локалізовувати різні додатки.

Все текстове наповнення зазвичай зберігається у відповідних мовних пакетах та файлах. Розглянемо деякі з найпоширеніших форматів файлів локалізації.

- .resx and .resw формат файлів використовується в Microsoft .NET;
- .resjson формат файлів використовується в Microsoft Windows 8/RT/Mobile;
- .properties формат файлів використовується в Java, FirefoxOS;
- .strings формат файлів використовується в Apple iOS;
- .yaml формат файлів використовується в Ruby/Rails;
- .ini формат файлів використовується в Joomla, PHP.

Кожен з цих форматів має свій синтаксис і для того щоб діставати потрібні дані необхідно мати спеціальний обробник.

В даному додатку планується локалізація цих форматів. Проте першим форматом який буде локалізуватись, буде саме Json. JSON в основному використовується для обміну даними між комп'ютерами і зберіганні інформації. Так як він має простий синтаксис і виділяє різні типи даних, він легко піддається для

обробки. Він має широку популярність і використовується для зберігання даних для локалізації у іграх, вебдодатках та комп'ютерних програмах.

Лістинг 2.1. Файл формату JSON

```
{
    "homePage": {
        "header": {
            "title": "Localiz",
            "signIn": "sign in",
            "signUp": "sign up"
        },
        "productHero": {
            "mainText": "Translate thanks to Localize",
            "infoText": "Use special packages that will
allow you to quickly localize.",
            "button": "register",
            "subText": "Discover the experience"
        }
}
```

Джерело: [створено автором]

Обробка файлів формату JSON проводитиметься за допомогою регулярних виразів. Регулярні вирази (англ. regular expressions) — формальна мова, яка використовується в комп'ютерних програмах, що працюють з текстом, для пошуку та здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволів [19]. Для того щоб ними скористатись .NET пропонує використати клас Regex.

Для вилучення строкових даних з файлу JSON був сформований наступний регулярний вираз:

$$([""'])(?:(?=(\\?))\2.)*?\1(.|\n) \qquad (2.1.)$$

Вираз (2.1.) дістає з файлу весь текст який обернений в лапки і ще один символ після лапок. На Рис. 2.6. показано як регулярний вираз виділяє потрібний текст.

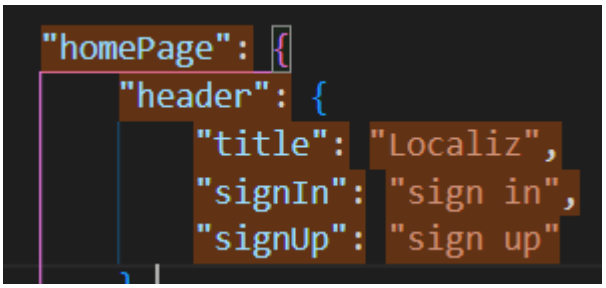


Рис. 2.6. Виділення тексту регулярним виразом

Джерело: [створено автором]

Також можна помітити, що регулярний вираз виділив назви значень, їх просто відсіяти так як з ними виділяється символ двокрапки.

Далі відібрані дані потрібно перенести в таблицю баз даних, де потім користувач буде проводити локалізацію.

База даних (Рис 2.7.) реалізована з використанням мови C#, а саме підходу CodeFirst для SQL Server. Цей підхід використовує можливості мови програмування C# для визначення структури та взаємозв'язків сутностей бази даних, а також автоматично генерує відповідний SQL-код для створення та управління схемою бази даних.

За допомогою підходу CodeFirst ми можемо визначати свої моделі баз даних за допомогою класів, властивостей та атрибутів C#, що представляють таблиці, стовпці та зв'язки між сутностями. Ці моделі слугують планом для схеми бази даних.

Використовуючи багаті можливості та гнучкість C#, ми можемо легко визначати складні зв'язки, такі як “один-до-одного”, “один-до-багатьох” та “багато-до-багатьох”. Підхід CodeFirst також підтримує успадкування, що дозволяє створювати ієрархії та поліморфні асоціації в межах схеми бази даних.

Після визначення моделі бази даних підхід CodeFirst дозволяє розробникам генерувати SQL-код, необхідний для створення бази даних та пов'язаних з нею таблиць, стовпців, обмежень та індексів. Цей процес, відомий як міграція бази даних, автоматично транслює моделі C# у SQL-оператори, забезпечуючи відповідність схеми бази даних визначеним моделям.

Крім того, підхід CodeFirst надає потужні інструменти та функції для управління змінами бази даних у часі. Ми можемо використовувати скрипти міграції для внесення оновлень до схеми бази даних без втрати існуючих даних, забезпечуючи плавну та контрольовану еволюцію структури бази даних.

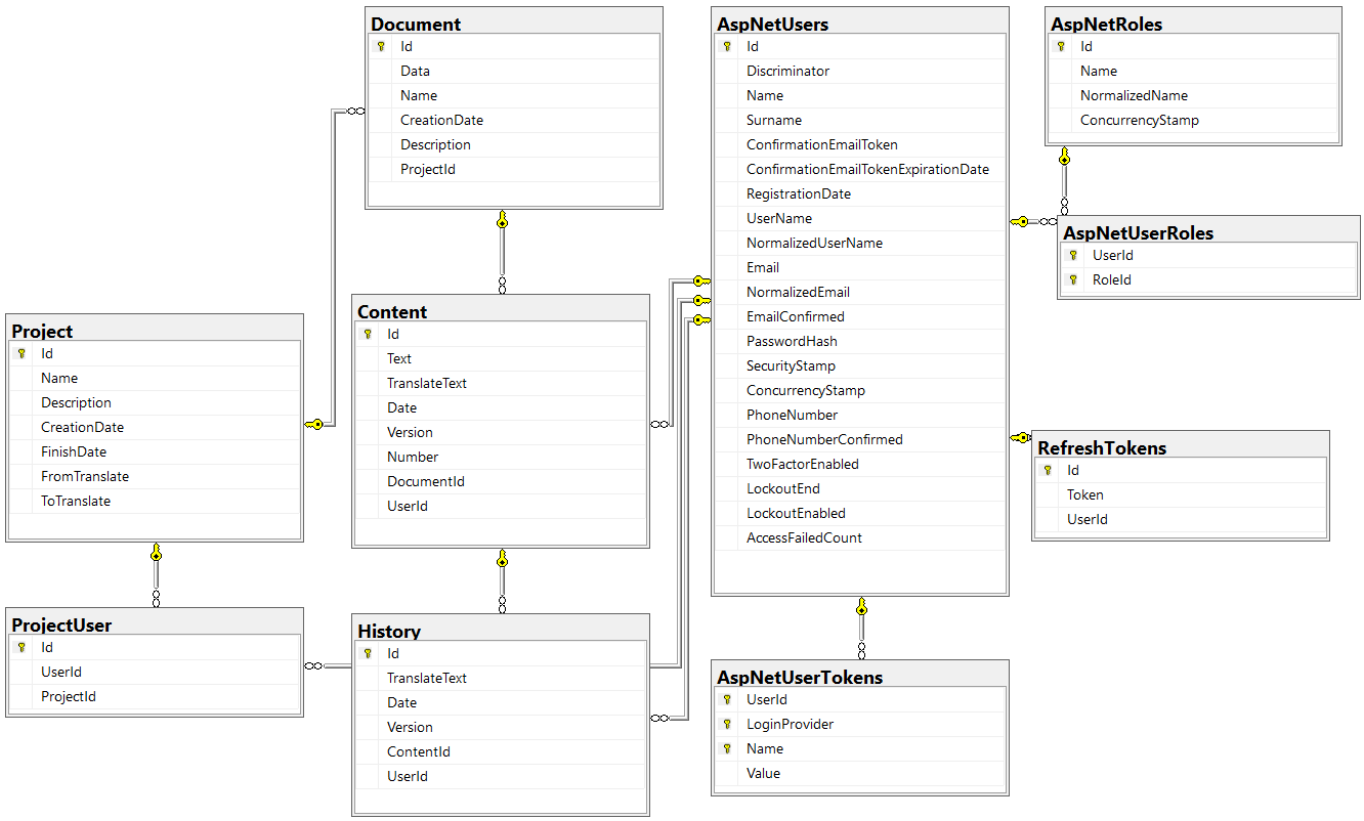


Рис. 2.7. База даних

Джерело: [створено автором]

Також попередньо була сформована USE-CASE діаграма, яка знаходиться в ДОДАТОК А

2.1.4. Сервіси серверної частини

Для зручного перегляду сервісів та контролерів використовується Swagger.

Почнемо з авторизації користувачів. Для захисту персональних даних та підвищення безпеки додатку використовується авторизація за допомогою Jwt (JSON Web Token) (Рис. 2.8.). Jwt - це компактний і самодостатній метод безпечної передачі

інформації між сторонами у вигляді JSON-об'єкта. Він відіграє вирішальну роль у перевірці автентичності та цілісності запитів користувачів в екосистемі додатку.

Використовуючи авторизацію Jwt, додаток гарантує, що тільки автентифіковані та авторизовані користувачі можуть отримати доступ до конфіденційних ресурсів і виконувати привілейовані дії. Коли користувач успішно входить в систему або надає дійсні облікові дані, сервер генерує токен Jwt, який потім безпечно передається клієнтській програмі.

Цей токен Jwt містить закодовану інформацію, таку як ідентифікатор користувача та відповідні дозволи, зашифровану за допомогою секретного ключа, відомого лише серверу. Щоразу, коли клієнтській програмі потрібно отримати доступ до захищеного ресурсу або виконати привілейовану операцію, вона включає токен Jwt в заголовок запиту.

Отримавши запит, сервер перевіряє автентичність токена Jwt, розшифровуючи і перевіряючи підпис за допомогою секретного ключа. Якщо токен дійсний і термін його дії не закінчився, сервер надає запитуваний доступ або виконує санкціоновану операцію. В іншому випадку він відмовляє в доступі і повертає відповідну відповідь про помилку.

Jwt-авторизація має кілька переваг для безпеки додатків. По-перше, вона усуває необхідність у сеансах на стороні сервера, зменшуючи витрати на зберігання та обробку даних. Крім того, оскільки токен має цифровий підпис, він забезпечує цілісність даних, що передаються, запобігаючи підробці або несанкціонованій модифікації. Крім того, використання шифрування в токенах Jwt захищає конфіденційну інформацію від несанкціонованого доступу.

Впроваджуючи авторизацію Jwt, додаток використовує надійний механізм безпеки, який підвищує захист персональних даних, посилює конфіденційність користувачів і знижує ризик несанкціонованого доступу або витоку даних.[1]

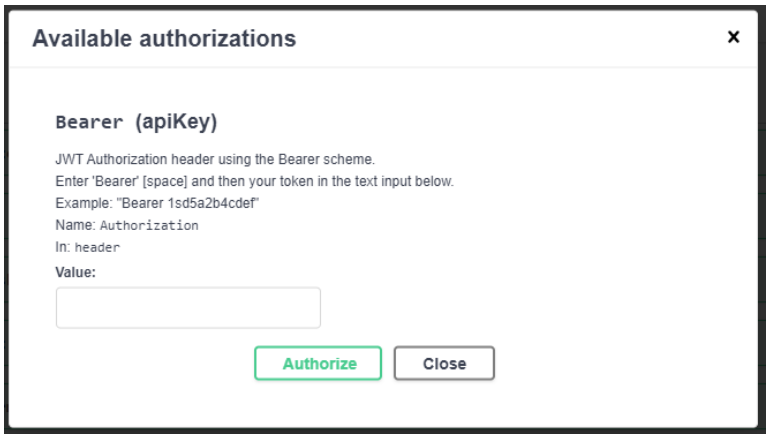


Рис. 2.8. Авторизація з допомогою JwtBearer

Джерело: [створено автором]

Не авторизований користувач може зареєструватися ввівши ім’я, фамілію, e-mail і пароль. Вже зареєстрований користувач, може увійти в додаток ввівши тільки e-mail і пароль. Також присутні контролер для refresh-token і для виходу з свого акаунта (Рис. 2.9.).

/api/Authentication/register – дозволяє зареєструватися для користувача.

/api/Authentication/login – дозволяє користувачу увійти вже створений акаунт

/api/Authentication/refresh-token – створює новий токен.

/api/Authentication/logout – дозволяє для користувача вийти з акаунту.

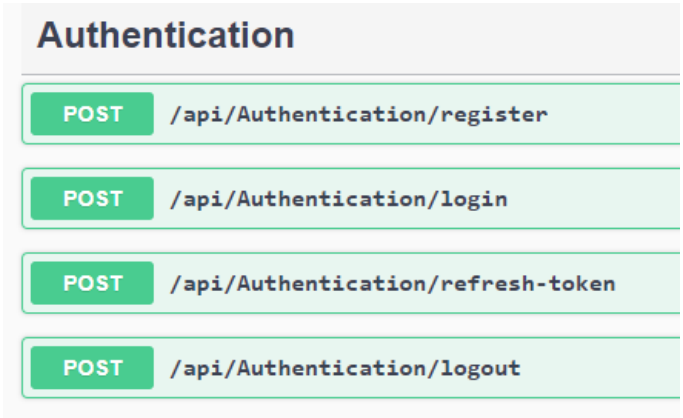


Рис. 2.9. Документація в Swagger контролера Authentication

Джерело: [створено автором]

Сервіси для роботи безпосередньо з файлом який локалізується (покищо можна локалізовувати файли розширення JSON). Перша можливість це саме завантаження файлу на вебдодаток, видалення, та повернення всіх користувачів які взаємодія з користувачем (Рис. 2.10.).

/api/Document/add – дозволяє користувачу додати файл до проєкту.

/api/Document – дозволяє користувачу видалити файл.

/api/Document/all-for-user – дозволяє користувачу побачити всю інформацію про цей файл.

/api/Document/download – завантажує перекладений файл на пристрій користувача.

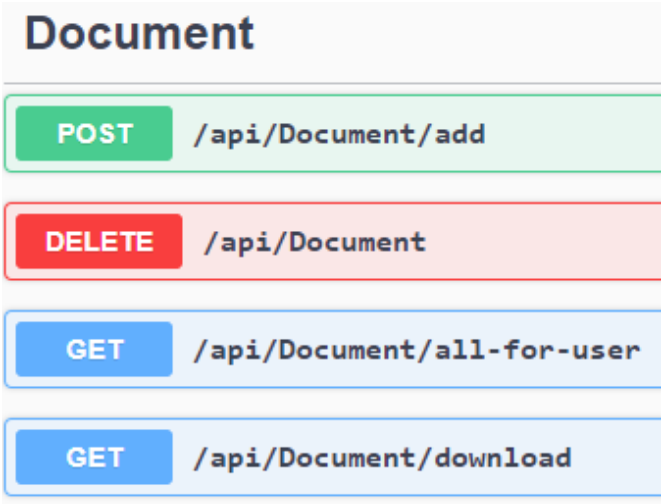


Рис. 2.10. Документація в Swagger контролера Document

Джерело: [створено автором]

Контент представляє собою всі стрічки які потрібно перекласти. Користувач може отримати для перегляду відповідну кількість елементів контенту. Для зручності перегляду була реалізована пагінація. Наступна можливість це переклад стрічки і третя можливість це переклад усього контенту документа (Рис. 2.11.).

/api/Content/get-range – дозволяє користувачу отримати частину історії.

/api/ Content /translate – дозволяє користувачу перекладати стрічки історії.

/api/Content/translate-document – застосовує машинний переклад до усього документа.

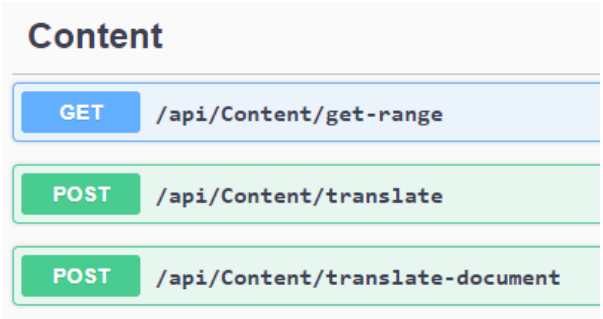


Рис. 2.11. Документація в Swagger контролера History

Джерело: [створено автором]

Для того щоб можна було додати файл потрібно створити проєкт. За це відповідають контролери Project (Рис. 2.12.).

/api/Project/add-user – дозволяє додавати користувачів до проекту.

/api/Project/create – дозволяє створювати проєкт.

/api/Project/all-for-user – дозволяє отримати всіх користувачів які мають доступ до проекта.

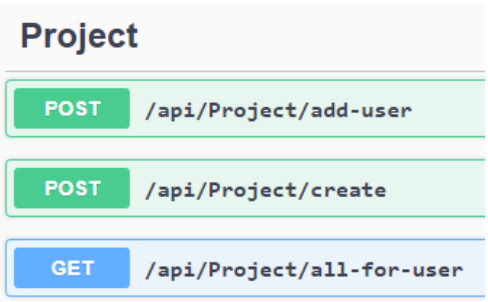


Рис. 2.12. Документація в Swagger контролера Project

Джерело: [створено автором]

Користувач може взаємодіяти із своїми персональними даними через контролери User (Рис. 2.13.).

/api/Users/user-info – повертає інформацію про юзера.

/api/Users/edit-info – дозволяє користувачу змінювати свої персональні дані.

/api/Users/user-full-name – повертає повне ім’я користувача.

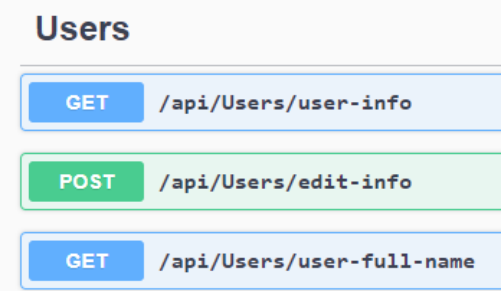


Рис. 2.13. Документація в Swagger контролера User

Джерело: [створено автором]

Історія дозволяє зберігати і переглядати усі зміни які зробили користувачі (Рис. 2.14.).

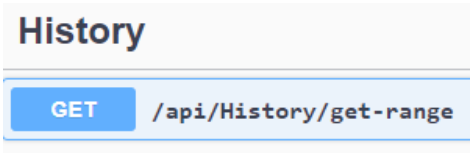


Рис. 2.14. Документація в Swagger контролера Project

Джерело: [створено автором]

2.1.5. Сервіси клієнтської частини

Програма створенна надійною та масштабованою за допомогою ReactJS.

Перш за все, налаштування середовища розробки для роботи над вебдодатком на основі React js. Перш ніж почати писати код, ось список інструментів, які використовувались під час розробки:

- Редактор коду – Visual Studio Code.
- Browser – Google Chrome.

На клієнтській частині реалізовано сторінк для гостей, яка складається з наступних частин:

AppAppBar (Рис. 2.15.) – хедер сайту на якому розміщуються кнопки зміни мови, вхід і реєстрація.

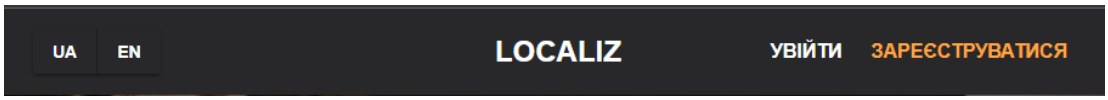


Рис. 2.15. Реалізація секції AppAppBar

Джерело: [створено автором]

ProductHero (Рис. 2.16.) – головна секція з короткою інформацією.

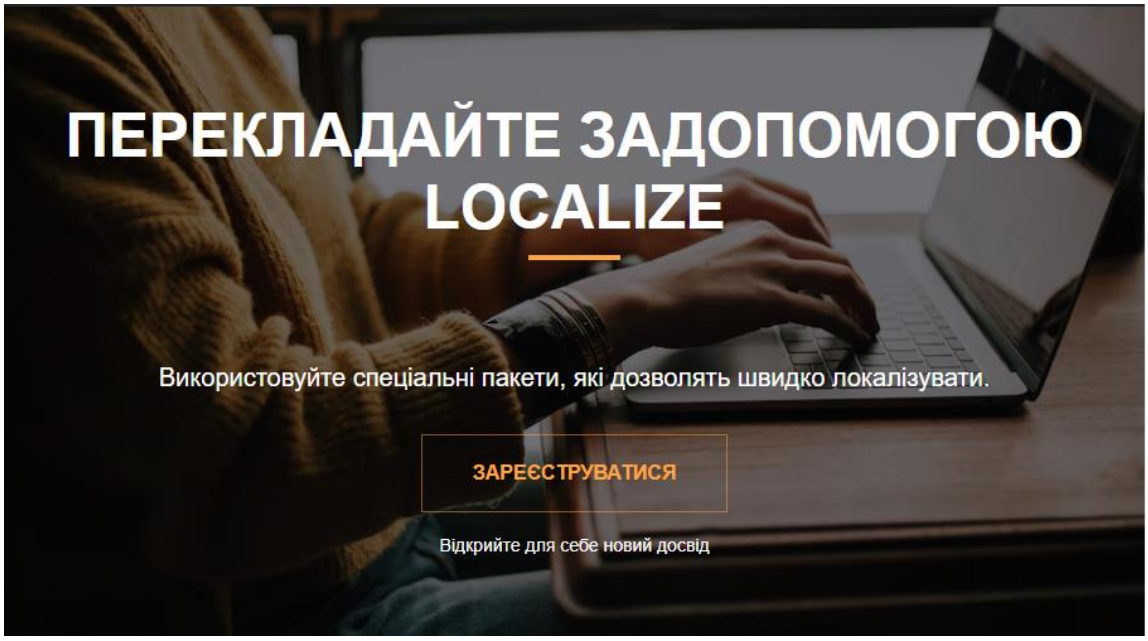


Рис. 2.16. Реалізація секції ProductHero

Джерело: [створено автором]

ProductValues (Рис. 2.17.) – секція коротким описом переваг додатку.

Рис. 2.17. Реалізація секції ProductValues

Джерело: [створено автором]

ProductCategories (Рис. 2.18.) – представлення можливостей і сервісів додатку.

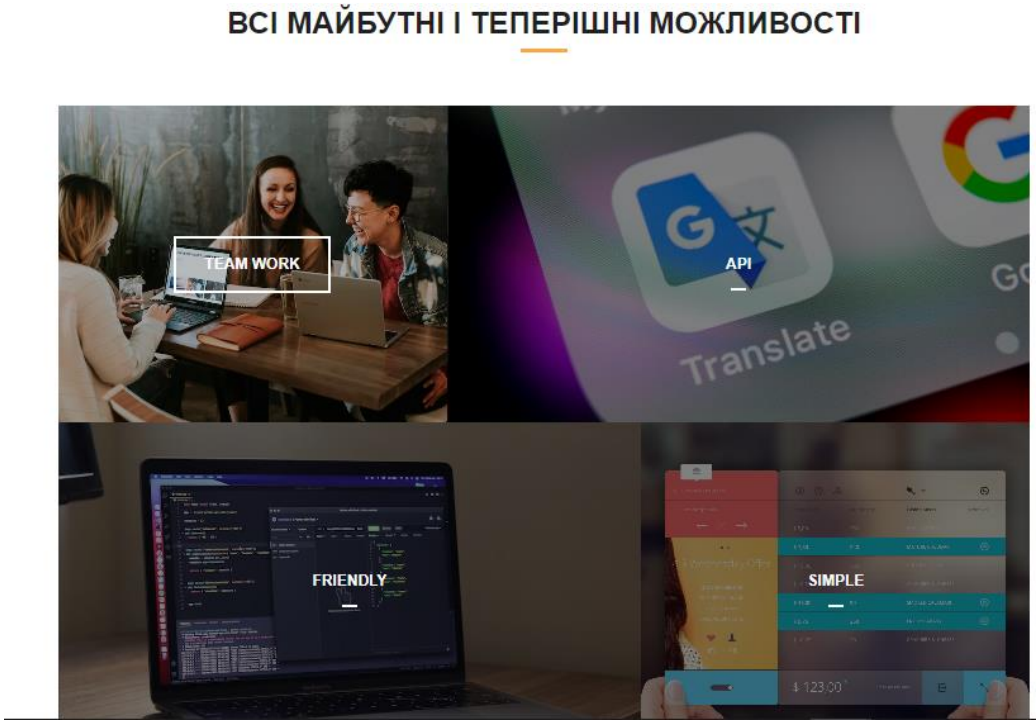


Рис. 2.18. Реалізація секції ProductCategories

Джерело: [створено автором]

ProductHowItWorks (Рис. 2.19.) – секція з коротким описом роботи з додатком.

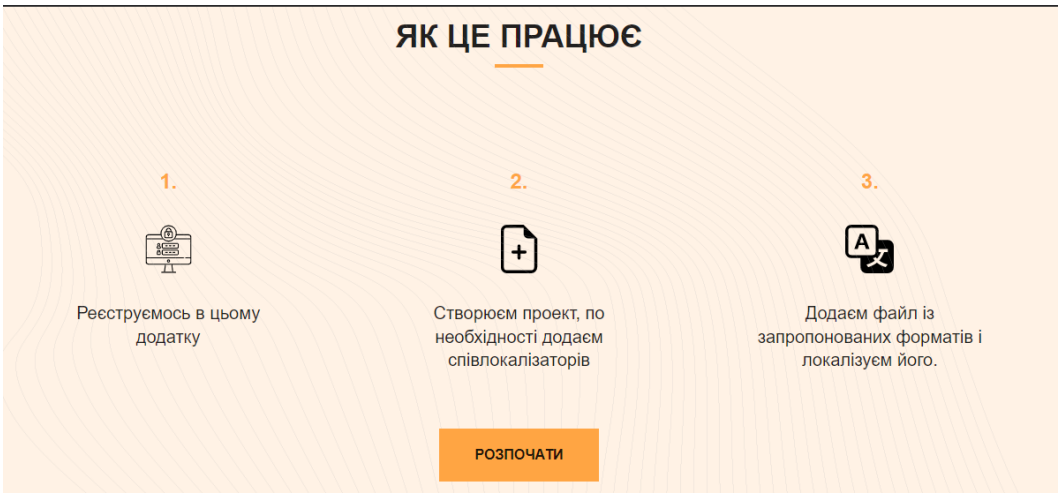


Рис. 2.19. Реалізація секції ProductHowItWorks

Джерело: [створено автором]

ProductCTA (Рис. 2.20) – секція, яка відповідає за відгуки про сайт.

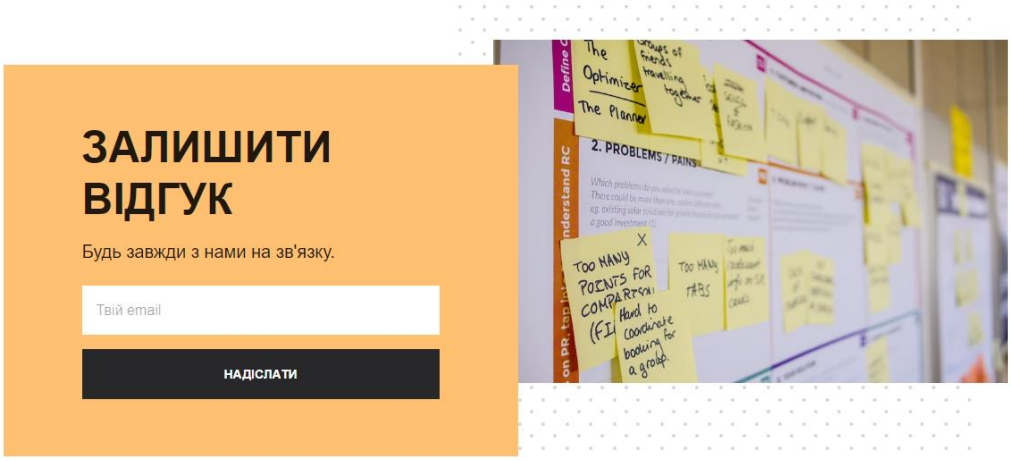


Рис. 2.20. Реалізація секції ProductCTA

Джерело: [створено автором]

ProductSmokingHero (Рис 2.21.) – секція зв’язку із розробником.

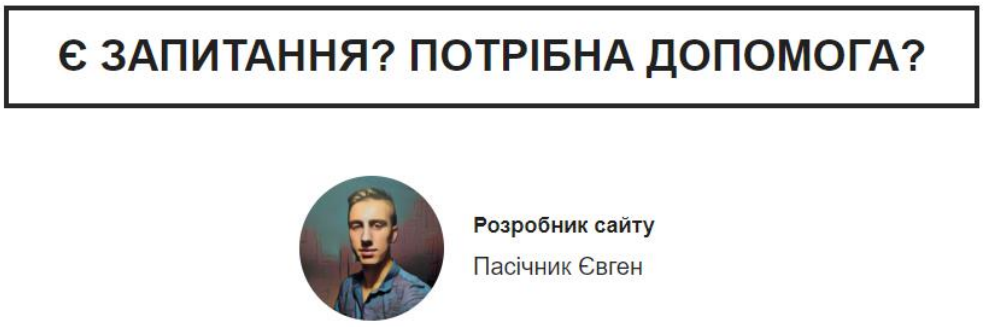


Рис. 2.21. Реалізація секції ProductSmokingHero

Джерело: [створено автором]

AppFooter (Рис. 2.22.) – секція футеру сайта.

Рис. 2.22. Секція AppFooter

Джерело: [створено автором]

Також присутні окремі сторінки реєстрації та авторизації (Рис. 2.23.).

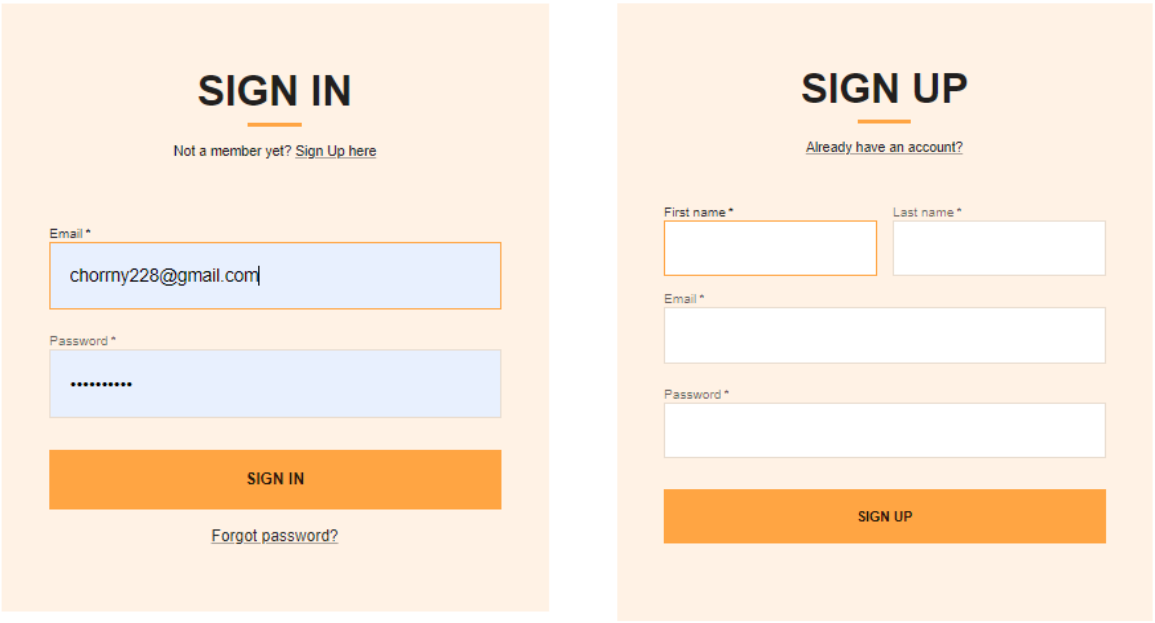


Рис. 2.23. Форми для авторизації та реєстрації

Джерело: [створено автором]

Ми розробили інтерфейс для зареєстрованого користувача, який дозволяє користувачеві перекладати файли формату .json. Інтерфейс розроблявся на базі готового та безкоштовного демонструє під назвою «material-kit-react».

Доступ до всього функціоналу вебдодатку здійснюватиметься через меню (Рис. 2.24.). На ньому розміщені кнопки:

«Dashboard» - сторінка з інформацією про додаток;

«Projects» - тут користувач може оглянути доступні йому проекти та створити нові;

«Account» - тут користувач може змінити персональні дані;

«Error 404» - тимчасова сторінка, яка демонструє вигляд помилки 404;

«Login» - кнопка, яка посилає на сторінку Логіну;

«Log out» - кнопка, яка виходить з акаунта користувача.

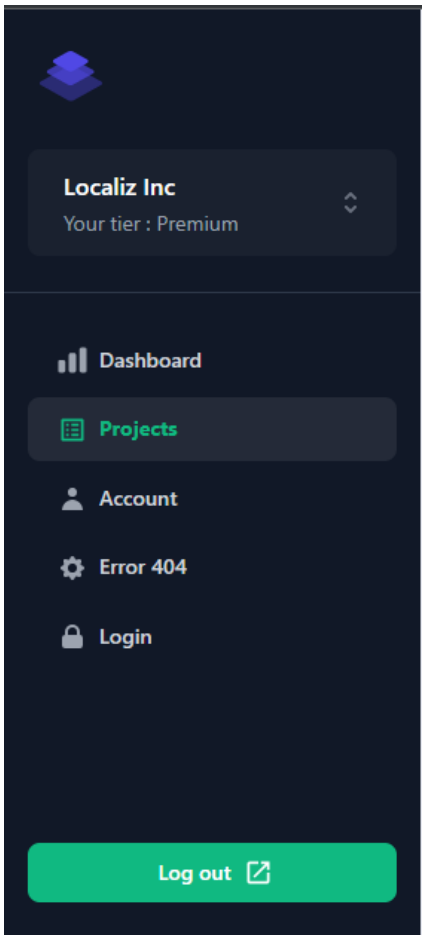


Рис. 2.24. Меню вебдодатка “Localiz”

Джерело: [створено автором]

Наступна частина інтерфейсу – це навбар. Він містить іконку користувача, і при кліку поній з’являється сайдбар (Рис. 2.25.) з іменем користувача і можливістю виходу з акаунта.

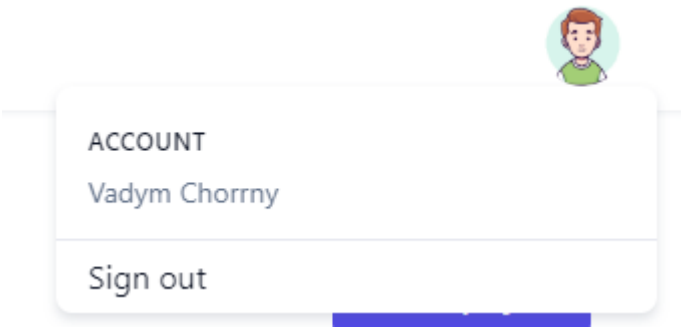


Рис. 2.25. Sidebar вебдодатка “Localiz”

Джерело: [створено автором]

На сторінці «Projects» (Рис. 2.26.) користувач може створювати нові проєкт, переглядати інформацію про них, по кліку на кнопку «Add co-interpreter» долучати інших користувачів до проекту. При натисканні на конкретний проєкт вас перене на сторінку проекта.

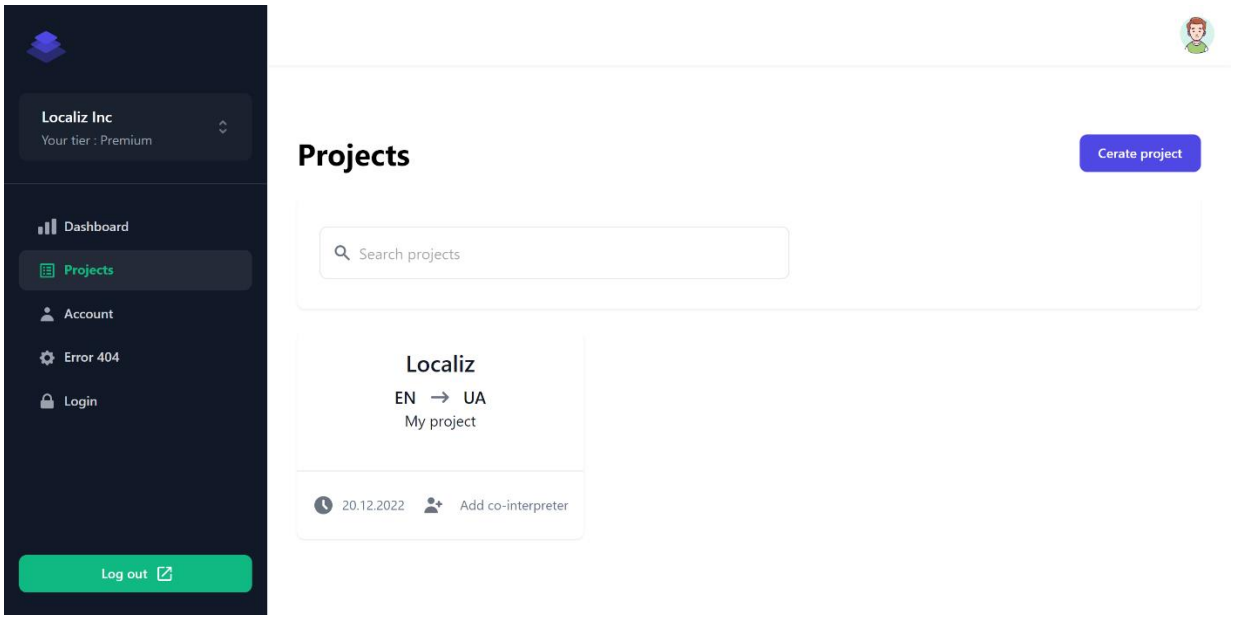


Рис. 2.26. Сторінка «Projects»

Джерело: [створено автором]

Для створення нового проекту використовується модальне вікно (Рис. 2.27.) з відповідними полями.

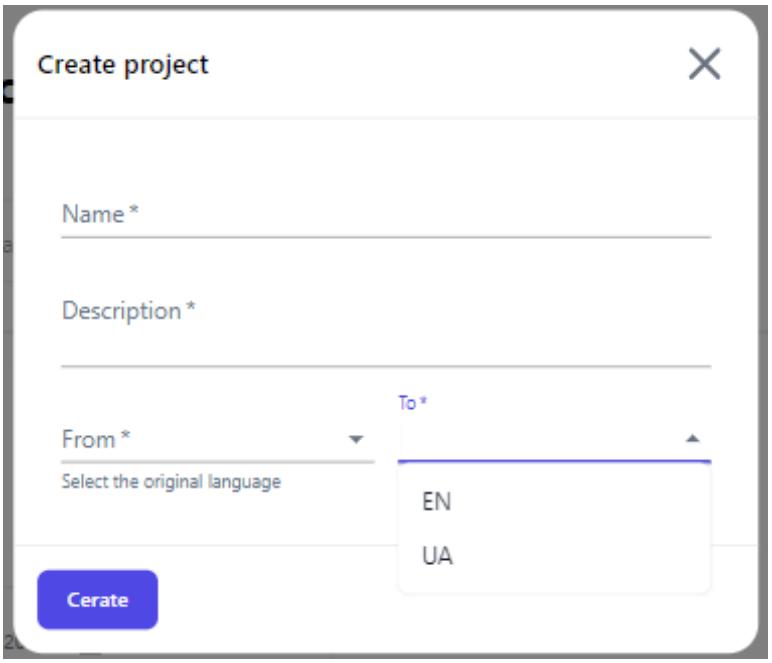


Рис. 2.27. Модальне вікно для створення проекту

Джерело: [створено автором]

Для долучення користувачів до проекту створено модальне вікно (Рис. 2.28.) з полем email.

Рис. 2.28. Модальне вікно для долучення користувачів

Джерело: [створено автором]

Сторінка проекта (Рис. 2.29.) містить таблицю з усіма файлами. Натиснувши на файл, користувач перейде на сторінку локалізації. Натиснувши на іконку корзини, файл видалиться. Кнопка "Add Document" відповідає за додавання нових документів до проекту.

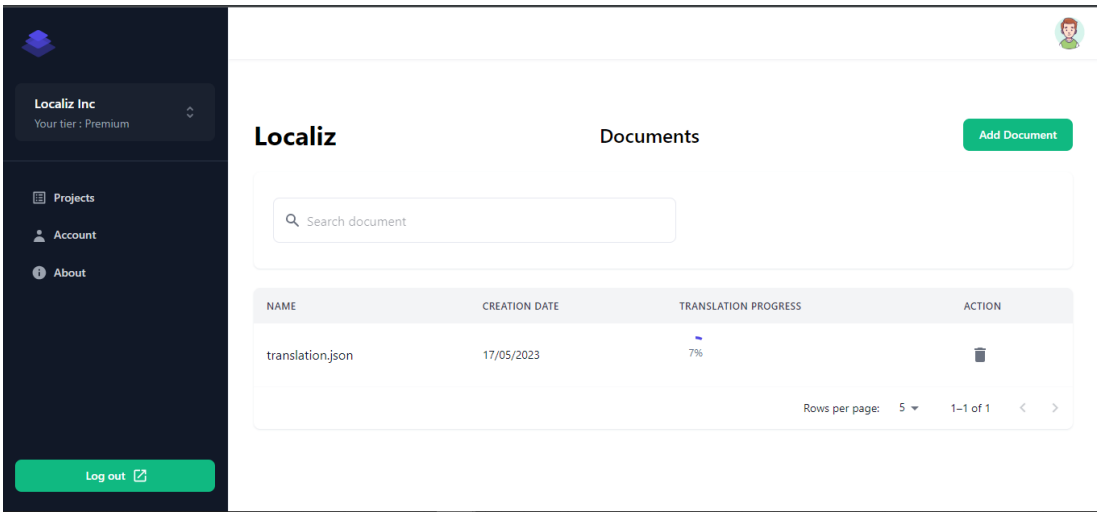


Рис. 2.29. Сторінка конкретного проекта

Джерело: [створено автором]

Сторінка локалізації (Рис. 2.28) містить таблицю з усіма стрічками тексту, які потрібно локалізувати. Відповідно, потрібно вписати текст перекладу і натиснути на іконку збереження. Також користувач може розгорнути і подивитися детально, хто і коли локалізував дану стрічку. Кнопка «Get document» повертає локалізований файл.

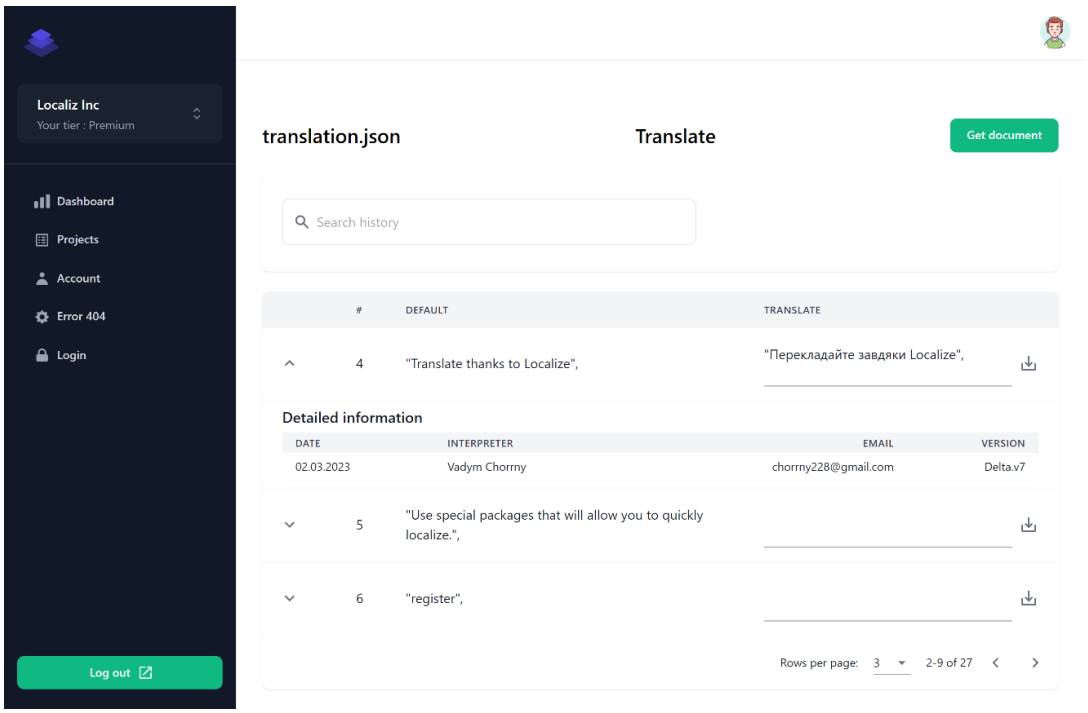


Рис. 2.30. Сторінка для перекладу документа

Джерело: [створено автором]

Сторінка про опис застосування додатка. В якому описуються всі необхідні дії та кроки з наведеними кроткими та зацикленими відео (Рис. 2.31.).

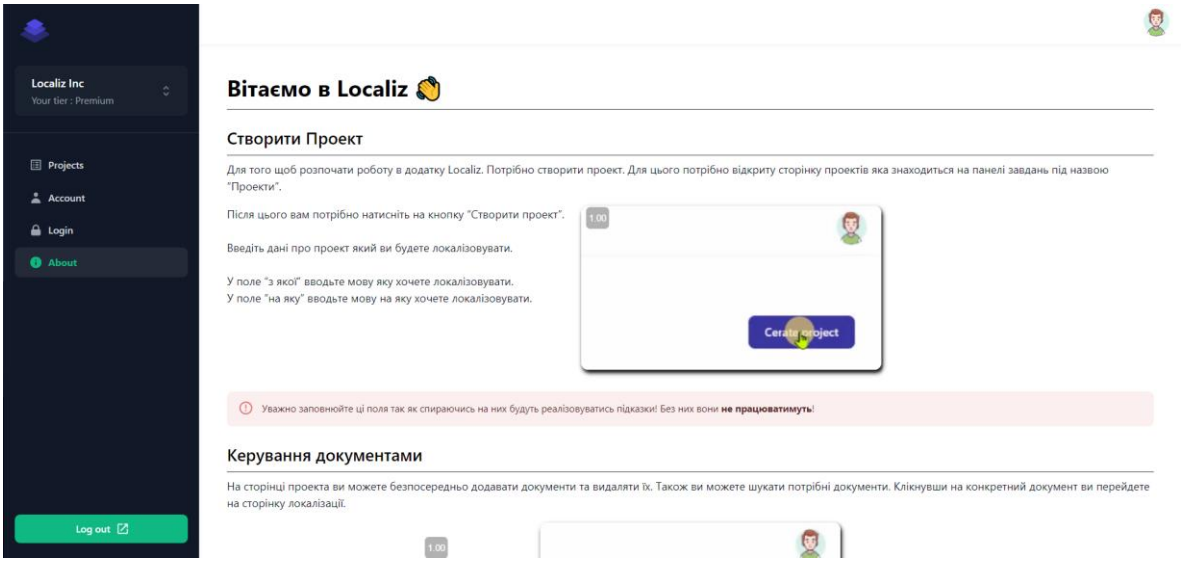


Рис. 2.31. Сторінка з інформацією про вебдодаток

Джерело: [створено автором]

2.2. Інтеграція стороннього API

Інтеграція API MyMemory - Translation Memory до нашого вебдодатоку, який дозволять програмі використовувати автоматичний машинний переклад.

Загальний огляд процесу інтеграції:

Отримання облікових даних API: Щоб використовувати MyMemory - Translate Memory API, ми повинні зареєструвати обліковий запис і отримати облікові дані API, які включають в себе ключ API. Це дозволить автентифікувати та авторизувати запити до API з нашого вебдодатку. Ми зарегіструвалися на сайті https://rapidapi.com/. Це зручний майданчик на якому розміщуються велика кількість API.

Огляд документації API Ознайомилися з документацією API MyMemory - Translation Memory.

Налаштування запитів до API: Налаштували запити API у нашому вебдодатку відповідно до інструкцій. Це передбачає створення HTTP-запитів з використанням відповідних HTTP-методів (запит GET search translations) і включення необхідних

параметрів, таких як вихідна і цільова мови, текст, який потрібно перекласти, а також ключ API для автентифікації.

Реалізація викликів API: Додали виклик API до коду нашої вебпрограми. Для цього використовували C# (для бекенд частини) і JavaScript (для фронтенд частини).

Обробка відповідей API: У нашому вебдодатку обробляєм відповіді API MyMemory - Translate Memory, які були нам надіслані. Це включає в себе обробку будь-яких проблем або винятків, які можуть виникнути під час запитів API.

Тестування та налагодження: Переконалися, що MyMemory - Translation Memory API належним чином інтегрований у наш вебдодаток, провівши ретельний тестовий запуск.

Безпеки: Щоб захистити конфіденційну інформацію та запобігти несанкціонованому доступу до API, впровадили необхідні процеси автентифікації та авторизації, зберігаєм облікові дані API в окремих файлах презначених для зберігання секретних даних.

Отже, ми успішно інтегрували API MyMemory - Translation Memory у вебдодаток, що включає кілька кроків, зокрема отримання облікових даних API, ознайомлення з документацією API, налаштування запитів до API, реалізацію викликів API, обробку відповідей API, тестування та налагодження, а також забезпечення заходів безпеки. Дотримуючись цих кроків ми отримуємо функціональність, що надається API і ефективно використовуєм у вебдодатку, забезпечуючи при цьому безпеку даних і автентифікацію.

2.3. Перспективи розвитку та впровадження додатку на ринок вебдодатків

Розробка вебдодатка з чітко поставленою метою та типом, полегшує роботу над проєктом. Важче стає тоді, коли справа доходить до просування та розповсюдження вебпрограми.

Розробка вебдодатку “Localiz” знаходиться на початку своєму шляху і основна перспектива - це доповнення додатка та формування його цілісності. Невід’ємна частина будь-якого вебдодатку є розміщення його на хостингу.

Наступна наша мета, безумовно, полягає в залученні користувачів і в швидкому відвідуванні та можливості легкого надання інформації нашим потенційним клієнтам. Саме через це методи просування не будуть такими ж, як для нативних або гібридних програм, які ми можемо знайти в магазинах.

Під час розробки вебдодатків доцільно знайти підсектор у межах нашої теми, щоб оптимізувати наші зусилля. Потрібні дослідження ринку і даного підсектора проведені в розділі «1.2. Аналіз продуктів-конкурентів на ринку».

Для того щоб залучати користувачів потрібно провести рекламну компанію. Найпотужнішим інструментом для розміщення онлайн-реклами наразі є Adwords або Adsense, обидва інструменти є пошуковими системами Google для розміщення платних оголошень. Зробити це можна в результатах пошукових систем, рекламному просторі з інших сайтів, відео тощо.

Це найшвидший спосіб просування нашої вебпрограми серед користувачів, які можуть бути зацікавлені у нашій вебпрограмі через пов’язаний вміст.

Однією з цілей вебпрограм є зробити їх доступними з мобільного пристрою. Доцільним є розробка адаптивного інтерфейсу, який расширить кількість потенційних користувачів. Також прийняте рішення розробки додатку на React позитивно впливає на його адаптивність, так як на його базі побудований фреймворк для мобільних додатків React Native.

Як ми можемо спостерігати перспективи в розробці веб додатку “Localiz” чи малі. Потрібно завершити весь основний функціонал, розмістити додаток на хостингу і реалізувати адаптацію.

Висновки до розділу 2

У розділі 2 було проведено поглиблене дослідження найпопулярніших архітектурних рішень як для бекенд-, так і для фронтенд-компонентів додатку, що охоплює весь проект. Процес розробки включав реалізацію серверних та клієнтських сервісів для забезпечення безперебійної функціональності. Крім того, було інтегровано сторонній API для машинного перекладу, що розширило можливості додатку. Також були ретельно прораховані майбутні перспективи програми на ринку веб-додатків.

Внутрішня частина програми була ретельно розроблена з використанням C# у поєднанні з .NET Core 6. Визнана однією з найпоширеніших мов програмування, C# пропонує сувору типізацію та об'єктно-орієнтовану парадигму. Вона може похвалитися потужною та розгалуженою спільнотою, яка активно сприяє її розвитку, що призвело до появи величезної кількості бібліотек та пакетів. Враховуючи ці переваги, вибір C# як основної мови для розробки програми Localiz був очевидним. У поєднанні з підходом чистої архітектури це дозволило створити надійну і легко підтримувану внутрішню інфраструктуру, що забезпечує довгострокову масштабованість і надійність.

З боку фронтенду було використано мову програмування JavaScript, зокрема React, для створення виняткового користувацького інтерфейсу. React вважається одним з найкращих варіантів для розробки інтерфейсу, оскільки надає розробникам потужні інструменти та ефективні робочі процеси. Для подальшого покращення процесу розробки були використані додаткові бібліотеки, такі як Material-UI та її шаблони. Незважаючи на початкові проблеми, що виникли на етапі розробки, всебічні ресурси та підтримка спільноти розробників з відкритим вихідним кодом були легкодоступними, що сприяло вирішенню будь-яких проблем, які виникали.

Крім того, інтеграція стороннього API, відомого як MyMemory - Translation Memory, значно розширила функціональність і зручність використання вебдодатку “Localiz”. Використання можливостей машинного перекладу, які надає цей API,

дозволило користувачам скористатися перевагами автоматизованих процесів перекладу, що ще більше спростило їхні зусилля з локалізації.

В кінці, під час аналізу перспектив додатку було враховано зростаючу важливість мобільної доступності. Враховуючи цю тенденцію, у майбутніх планах було передбачено виведення додатка “Localiz” на ринки мобільних додатків, що забезпечить безперешкодний доступ до мобільних пристроїв. Використовуючи широку доступність мобільних платформ, “Localiz” зможе розширити свою базу користувачів і задовольнити потреби своєї цільової аудиторії.

ВИСНОВКИ

Під час виконання даної кваліфікаційної робити ми розробили сервісну і клієнтську частини вебдодатку «Localiz». Також ми реалізували всі необхідні сервіси для функціонування вебпрограми.

Це дослідження успішно досягло своєї мети - спроектувало та розробило програмний продукт, вебдодаток “Localiz”, який надає цінні інструменти та ресурси для реалізації перекладу текстового контенту в межах програмних додатків. Завдання дослідження були ефективно вирішені завдяки комплексному підходу, який охоплював різні аспекти процесу локалізації.

У ході роботи ми проаналізували та дослідили тему локалізації програмних додатків і обрали сферу за яку буде відповідати наш додаток, а саме локалізацію тексту. Провили анілз ринку і виявили основних конкурентів та проаналізували їх і на цих знаннях сформулювали SWOT-аналіз.

Ми дослідили і обрали оптимальне архітектурне рішення для нашого додатку, а саме Чиста архітектура. Серверну частину реалізували на .NET з такими технологіями як:

- ASP.NET Core 6.
- Jwt Authorization.
- Ardalis.Specification.EntityFrameworkCore та інші …

Реалізацію клієнтської частини ми реалізовували на React з такими технологіями:

- React.
- Redux.
- Material UI та інші …

Всі частини проекту ми розмістили на Github. Цей застосунок надзвичайно спрощує розробку IT-проекта.

При розробці ми зустрічалися з різними труднощами, які в основному пов’язані з розробкою клієнтської частини, а саме проблема з версіями деяких бібліотек, таких як Material UI і React. Material UI має потужній функціонал для створення інтерфейсу, але на пряму відмовляється працювати з новою версією React. Ці проблеми доволі сильно сповільнювали розробку.

Так як локалізація додатка може зберігатися в різних типах файлів, ми вирішили першим зробити локалізацію JSON файлів. Для виконання цієї технології було використано клас Regex, який надає змогу завдяки метасимволів, працювати з текстом, для пошуку та здійснення маніпуляцій з підрядками в тексті.

Створили базу даних на SQL за допомогою CodeFirst підходу. Реалізували контроли які надають основний функціонал.

Так як наша вебпрограма має мати змогу колективної розробки була створена авторизація, яка побудована на основі Jwt Bearer. Це надасть деякий захист для користувачів від зловмисників, так як ми не працюємо прямо з даними для авторизації, а формуємо токен і шифруєм паролі.

Користувач може створювати свої проекти для локалізації додавати туди файли типу JSON, користувачів, які будуть мати доступ до нього. Також ми додали пагінацію, яка дозволить раціонально формувати запити для отримання стрічок які потрібно локалізувати.

Крім того, ми інтегрували API MyMemory - Translation Memory у наш вебдодаток, виконавши низку кроків, які дозволили нам ефективно використовувати його функціональність. Ми отримали облікові дані API шляхом реєстрації облікового запису та отримання ключа API, переглянули документацію API, налаштували запити API, реалізували виклики API, обробили відповіді API, а також протестували та налагодили інтеграцію. Налаштували його для ефективного і безпечного використання.

Все це сформулювало міцний фундамент для нашого додатку. До повного завершення роботи ще потрібно зробити:

- розширити функціонал вебдодатку;
- розмістити проєк на хостингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Using JwtBearer Authentication in an API-only ASP.NET Core Project URL: https://wildermuth.com/2018/04/10/Using-JwtBearer-Authentication-in-an-API-only-ASP-NET-Core-Project/. (дата звернення: 20.03.2023 р.).

2. React JS – Architecture + Features + Folder structure + Design Pattern URL: https://medium.com/geekculture/react-js-architecture-features-folder-structure-design-pattern-70b7b9103f22. (дата звернення: 20.03.2023 р.).

3. Entity Framework Core URL: https://learn.microsoft.com/ru-ru/ef/core/. (дата звернення: 20.03.2023 р.).

4. Shvets A. Refactoring Guru URL: https://refactoring.guru/uk/design-patterns. (дата звернення: 20.03.2023 р.).

5. George Swagger For .NET MVC Web API URL: https://www.c-sharpcorner.com/article/swagger-for-net-mvc-web-api/. (дата звернення: 20.03.2023 р.).

6. Bogard J. AutoMapper URL: https://www.simform.com/blog/web-application-architecture/. (дата звернення: 20.03.2023 р.).

8. Dhaduk H. An Ultimate Guide to Web Application Architecture URL: https://www.simform.com/blog/web-application-architecture/. (дата звернення: 20.03.2023 р.).

7. ElHady H. Top Mobile App Localization Tools URL: https://blog.instabug.com/mobile-app-localization-tools/ (дата звернення: 20.12.2022 р.).

8. Martin R. The Clean Code Blog URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. (дата звернення: 20.03.2023 р.).

9. McMaster K. Getting Started With Specifications URL: https://blog.nimblepros.com/blogs/getting-started-with-specifications/. (дата звернення: 20.03.2023 р.).

10. Phrase Software Localization. URL: https://phrase.com/blog/posts/software-localization/ (дата звернення: 20.12.2022 р.).

11. Iqbal M. App Download Data URL: https://www.businessofapps.com/data/app-statistics/#2.2. (дата звернення: 20.03.2023 р.).

12. Smith A. App Download Data URL: https://www.investopedia.com/articles/07/invest_china.asp. (дата звернення: 20.12.2022 р.).

13. Smith S. Architecting Modern Web Applications with ASP.NET Core and Azure. – Washington, 2022. 119 с.

14. Tutorial: Create a web API with ASP.NET Core URL: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio. (дата звернення: 20.12.2022 р.).

15. Tutorial: Create a .NET class library using Visual Studio URL: https://learn.microsoft.com/en-us/dotnet/core/tutorials/library-with-visual-studio?pivots=dotnet-6-0. (дата звернення: 20.12.2022 р.).

16. Peterson R. What is SQL Server? Introduction, History, Types, Versions URL: https://www.guru99.com/sql-server-introduction.html. (дата звернення: 20.03.2023 р.).

17. Monte L. Sweetalert2 URL: https://sweetalert2.github.io/#examples. (дата звернення: 20.03.2023 р.).

18. Keniston K. SOFTWARE LOCALIZATION: NOTES ON TECHNOLOGY AND CULTURE URL: http://web.mit.edu/~kken/Public/papers1/Software%20localization.htm. (дата звернення: 20.12.2022 р.).

19. Regex Класс [G. Warren, M. Wenzel, C. Sanchez та ін.]. URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex?view=net-6.0. (дата звернення: 20.12.2022 р.).

20. React URL: https://uk.legacy.reactjs.org/docs/getting-started.html (дата звернення: 20.12.2022 р.).

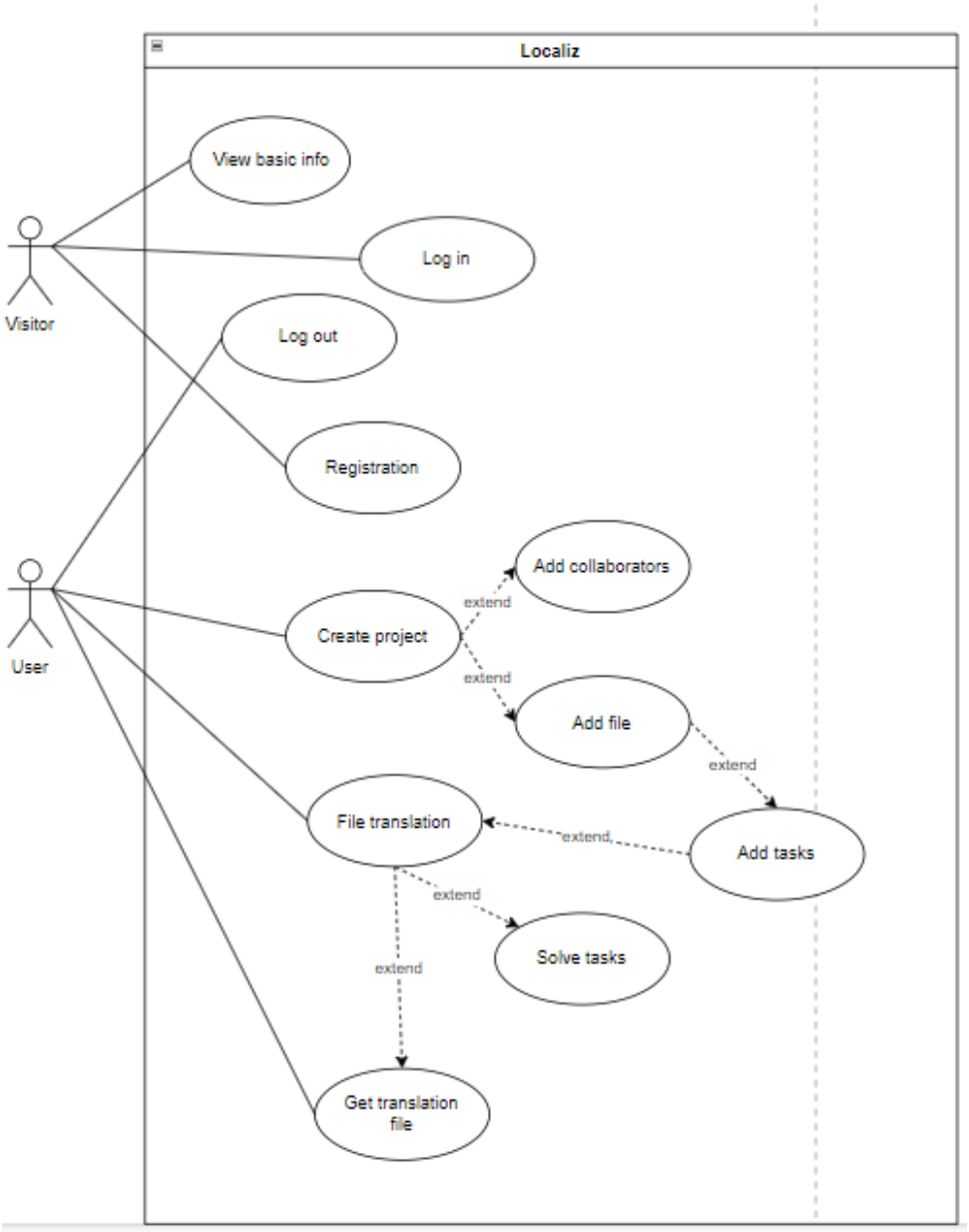
21. Material-UI URL: https://mui.com/toolpad/getting-started/overview/ (дата звернення: 20.12.2022 р.).

22. i18next documentation URL: https://www.i18next.com/ (дата звернення: 20.12.2022 р.).

ДОДАТКИ

ДОДАТОК А

USE CASE діаграма проєкту “localiz” процесу взаємодії користувачем із додатком



ДОДАТОК Б

Приклад реалізації патерну Repository

```
public class Repository<TEntity>
        : RepositoryBase<TEntity>, IRepository<TEntity> where TEntity : class
    {
        private ApplicationDbContext _context;
        private DbSet<TEntity> _dbSet;

        public Repository(ApplicationDbContext context) : base(context)
        {
            _context = context;
            _dbSet = context.Set<TEntity>();
        }

        public async Task<IEnumerable<TEntity>> AddRangeAsync(IEnumerable<TEntity>
entities)
        {
            await _dbSet.AddRangeAsync(entities);
            await _context.SaveChangesAsync();

            return entities;
        }

        public async Task UpdateRangeAsync(IEnumerable<TEntity> entities)
        {
            _dbSet.UpdateRange(entities);
            await _context.SaveChangesAsync();
        }
    }
```

ДОДАТОК В

Приклад реалізації API контролера для класу Document

```
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
[Route("api/[controller]")]
[ApiController]
public class DocumentController : ControllerBase
{
    private readonly IDocumentService _documentService;
    public DocumentController(IDocumentService documentService)
    {
        _documentService = documentService;
    }

    [HttpPost("add")]
    [AuthorizeByRole(IdentityRoleNames.User)]
    public async Task<List<AddByteDocDTO>> AddDocument([Required] uint
projectId,[Required] List<IFormFile> documents)
    {
        return await _documentService.AddDocument(projectId, documents);
    }

    [HttpDelete]
    [AuthorizeByRole(IdentityRoleNames.User)]
    public async Task DeleteDocument(uint documentId)
    {
        await _documentService.DeleteDocument(documentId);
    }

    [HttpGet("all-for-user")]
    [AuthorizeByRole(IdentityRoleNames.User)]
    public async Task<List<DocumentDTO>> GetAllDocuments(uint projectId)
    {
        return await _documentService.GetAllDocuments(projectId);
    }}
```

ДОДАТОК Г

Приклад реалізації Специфікацій для класу Document

```
public class DocumentSpecification
{
    internal class GetDocumentByProjectId : Specification<Document>
    {
        public GetDocumentByProjectId(uint projectId)
        {
            Query.Where(p => p.ProjectId == projectId);
        }
    }

    internal class GetDocumentWithHistories
        : Specification<Document>, ISingleResultSpecification<Document>
    {
        public GetDocumentWithHistories(uint documentId)
        {
            Query.Where(p => p.Id == documentId)
                .Include(d => d.Histories);
        }
    }
}
```

ДОДАТОК Д

Приклад реалізації компонента ProductHero за допомогою Material UI

```
import * as React from 'react';
import Button from '../components/Button';
import Typography from '../components/Typography';
import {Typography_h2} from '../components/Typography';
import { useTranslation } from 'react-i18next';
import ProductHeroLayout from './ProductHeroLayout';

const backgroundImage =
  'https://images.unsplash.com/photo-1515378960530-7c0da6231fb1?ixlib=rb-
4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=870
&q=80';

export default function ProductHero() {
  const { t, i18n } = useTranslation();

  const changeLanguage = (language) => {
    i18n.changeLanguage(language)
  }

  return (
    <ProductHeroLayout
      sxBackground={{
        backgroundImage: `url(${backgroundImage})`,
        backgroundColor: '#7fc7d9', // Average color of the background image.
        backgroundPosition: 'center',
      }}
    >
      <Typography_h2 color="inherit" align="center" variant="h2" marked="center">
        {t('homePage.productHero.mainText')}
      </Typography_h2>
      <Typography
        color="inherit"
        align="center"
        variant="h5"
        sx={{ mb: 4, mt: { sx: 4, sm: 10 } }}
      >
       {t('homePage.productHero.infoText')}
      </Typography>
      <Button
        color="secondary"
        variant="outlined"
```

ДОДАТОК Д

Приклад реалізації компонента ProductHero за допомогою Material UI

(продовження)

```
        size="large"
        component="a"
        href="/sign-up/"
        sx={{ minWidth: 200}}
      >
        {t('homePage.productHero.button')}
      </Button>
      <Typography variant="body2" color="inherit" sx={{ mt: 2 }}>
      {t('homePage.productHero.subText')}
      </Typography>
    </ProductHeroLayout>
  );
}
```